Arm® CoreLink™ MMU-700 System Memory Management Unit

Revision: r0p1

Technical Reference Manual

arm

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
101542_0001_04_en

Arm® CoreLink™ MMU-700 System Memory Management Unit

Technical Reference Manual

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-01	25 October 2019	Confidential	First issue for r0p0 BET release
0000-02	30 March 2020	Confidential	First issue for r0p0 LAC release
0001-03	14 September 2020	Non-Confidential	First issue for r0p1 EAC release
0001-04	19 February 2021	Non-Confidential	Second issue for r0p1 EAC release

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at *http://www.arm.com/company/policies/trademarks*.

Copyright © 2019–2021 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Additional Notices

Some material in this document is based on IEEE 754-2008 IEEE Standard for Binary Floating-Point Arithmetic. The IEEE disclaims any responsibility or liability resulting from the placement and use in the described manner.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

developer.arm.com

Progressive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

This document includes terms that can be offensive. We will replace these terms in a future issue of this document.

If you find offensive terms in this document, please contact *terms@arm.com*.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Contents
Arm® CoreLink™ MMU-700 System Memory Management Unit Technical Reference Manual

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Preface

This preface introduces the *Arm® CoreLink™ MMU-700 System Memory Management Unit Technical Reference Manual*.

It contains the following:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

About this book

This book is for the Arm® CoreLink™ MMU-700 System Memory Management Unit.

Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*x* Identifies the major revision of the product, for example, r1.

p*y* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the MMU-700.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter provides an overview of the MMU-700 *System Memory Management Unit* (SMMU).

Chapter 2 Functional description
This chapter describes the functionality of the MMU-700.

Chapter 3 Programmers model
This chapter describes the MMU-700 programmers model.

Appendix A Signal descriptions
This appendix describes the MMU-700 external signals.

Appendix B ELA signal descriptions
This section describes the **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals of the TCU and TBU components that are used to interface with external ELA.

Appendix C Software initialization examples
This appendix provides examples of how software can initialize and enable the MMU-700.

Appendix D Revisions
This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic
Introduces special terminology, denotes cross-references, and citations.

bold
Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

`mono`space
: Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
: Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
: Denotes language keywords when used outside example code.

`<and>`
: Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

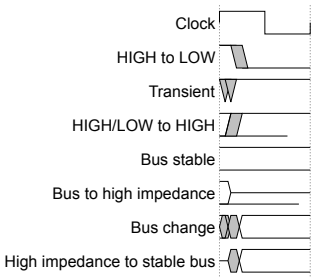

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level
: The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n
: At the start or end of a signal name, n denotes an active-LOW signal.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Additional reading

Arm publications

This document contains information that is specific to this product. See the following documents for other relevant information:

Table 1 Arm publications

Document name	Document ID	Licensee only
Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual	101543	Y
Arm® CoreLink™ MMU-700 System Memory Management Unit Release Note	PJDOC-1779577084-28963	Y
Arm® CoreLink™ LPD-500 Low Power Distributor Technical Reference Manual	100361	N
Arm® CoreLink™ LPD-500 Low Power Distributor Integration and Implementation Manual	100362	Y
Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual	100806	N
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	N
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual	101089	Y
Arm® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide	DUI 0615	Y
Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2	IHI 0070C.a	N
Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile	DDI 0487E.a	N
AMBA® AXI and ACE Protocol Specification	IHI 0022H	N
AMBA® 4 AXI4-Stream Protocol Specification	IHI 0051A	N
AMBA® APB Protocol Specification	IHI 0024C	N
AMBA® DTI Protocol Specification	IHI 0088E	N
AMBA® LTI Protocol Specification	IHI 0089A	N
AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces	IHI 0068C	N
Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A	DDI 0598B.a	N
Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A	DDI 0587C.b	N
Arm® Server Base System Architecture 7.0 Platform Design Document	DEN 0029	N

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm CoreLink MMU-700 System Memory Management Unit Technical Reference Manual*.
- The number 101542_0001_04_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Chapter 1
Introduction

This chapter provides an overview of the MMU-700 *System Memory Management Unit* (SMMU).

It contains the following sections:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.1 About the CoreLink™ MMU-700 System Memory Management Unit

The MMU-700 is a *System*-level *Memory Management Unit* (SMMU) that translates an input address to an output address. This translation is based on address mapping and memory attribute information that is available in the MMU-700 internal registers and translation tables.

The MMU-700 implements the Arm SMMU architecture version 3.2, SMMUv3.2, as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

An address translation from an input address to an output address is described as a stage of address translation. The MMU-700 can perform:

- Stage 1 translations that translate an input *virtual address* (VA) to an output *physical address* (PA) or *intermediate physical address* (IPA)
- Stage 2 translations that translate an input IPA to an output PA
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The MMU-700 performs translation table walks for each stage of the translation.

In addition to translating an input address to an output address, a stage of address translation also defines the memory attributes of the output address. With a two-stage translation, the stage 2 translation can modify the attributes that the stage 1 translation defines. A stage of address translation can be disabled or bypassed, and the MMU-700 can define memory attributes for disabled and bypassed stages of translation.

The MMU-700 uses inputs from the requesting master to identify a context. Configuration tables in memory define how the MMU-700 is to translate each context, such as which translation tables to use.

The MMU-700 can cache the result of a translation table lookup in a *Translation Lookaside Buffer* (TLB). It can also cache configuration tables in a configuration cache.

The MMU-700 contains the following key components:

- *Translation Buffer Units* (TBUs) that use a TLB to cache translation tables
- A *Translation Control Unit* (TCU) that controls and manages address translations
- *Distributed Translation Interface* (DTI) interconnect components that connect multiple TBUs to the TCU

Related concepts

2.1 About the functions on page 2-23

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.2 Compliance

The MMU-700 complies with, or implements, the specifications that this section describes. This *Technical Reference Manual* (TRM) complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

1.2.1 Arm architecture

The MMU-700 implements parts of the Armv8.4 *Virtual Memory System Architecture* (VMSA), as the *Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile* defines. The SMMUv3.2 architecture describes the parts of VMSA that apply to the MMU-700.

1.2.2 SMMU architecture

The MMU-700 implements the SMMUv3.2 architecture, as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

Related concepts

2.4.1 SMMUv3 implementation on page 2-57

1.2.3 AMBA Distributed Translation Interface protocol

The MMU-700 implements the *Distributed Translation Interface* (DTI) protocol, as the *AMBA® DTI Protocol Specification* defines.

The DTI interfaces use an AXI4-Stream interface, as the *AMBA® 4 AXI4-Stream Protocol Specification* defines.

Related concepts

2.3.1 DTI overview on page 2-40

1.2.4 AMBA ACE5-Lite and AMBA AXI5 protocol

The MMU-700 complies with the AMBA ACE5-Lite protocol.

For more information, see the *AMBA® AXI and ACE Protocol Specification*.

Related concepts

2.4.2 AMBA implementation on page 2-60

1.2.5 AMBA APB protocol

The MMU-700 complies with the AMBA APB4 protocol, as the *AMBA® APB Protocol Specification* defines.

1.2.6 LTI protocol

The MMU-700 complies with the LTI protocol, as the *AMBA® LTI Protocol Specification* defines.

Related concepts

LTI TBU LTI interface on page 2-36

1.2.7 LPI Q-Channel protocol

The MMU-700 complies with the LPI Q-Channel, as the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces* defines.

Related references

A.1.6 TCU LPI_PD interface signals on page Appx-A-174

A.1.7 TCU LPI_CG interface signals on page Appx-A-174

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.2.5 TBU LPI_PD interface signals on page Appx-A-190
A.2.6 TBU LPI_CG interface signals on page Appx-A-191

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.3 Features

The MMU-700 provides the following features:

Compliance with the SMMUv3.2 architecture

- Support for stage 1 translation, stage 2 translation, and stage 1 followed by stage 2 translation
- Support for Armv8 AArch32 and AArch64 translation table formats
- Support for 4KB, 16KB, and 64KB granule sizes in AArch64 format
- Support for *PCI Express* (PCIe) integration, including:
  - *Address Translation Services* (ATS), including full and split-stage ATS
  - *Process Address Space IDs* (PASIDs)
  - *Access Control Services* (ACS)
- Support for *Page Request Interface* (PRI), as SMMUv3 defines. PRI is an optional PCIe ATS extension that enables support for unpinned memory in PCIe.
- Support for MPAM
- Support for Secure-EL2
- Masters can be stalled while a processor handles translation faults, enabling software support for on-demand paging
- Configuration tables in memory can support more than a million active translation contexts
- Queues in memory perform MMU-700 management. There is no requirement to stall a processor when it accesses the MMU-700.
- A *Performance Monitoring Unit* (PMU) in each TBU and TCU that enables MMU-700 performance to be investigated
- *Reliability, Serviceability, and Availability* (RAS) features for RAM corruption detection and correction

Support for AMBA interfaces

- ACE5-Lite TBU transaction interfaces that support cache stash transactions, deallocating transactions, and cache maintenance
- An architected AXI5 extension that communicates per-transaction translation stream information
- An ACE5-Lite+*Distributed Virtual Memory* (DVM) TCU table walk interface that enables Armv8.5 processors to perform shared TLB invalidate operations without accessing the MMU-700 directly
- An ACE5 Low-Power extension that enables the TCU to subscribe to DVM TLB invalidate requests on powerup and powerdown without reprogramming the DTI interconnect
- AMBA DTI communication between the TCU and TBUs, enabling masters to request translations and implement TBU functionality internally
- Support for the AMBA *Low-Power Interface* (LPI) Q-Channel so that standard controllers can control power and clock gating
- AXI5 **WAKEUP** signaling on all interfaces, including DTI and APB interfaces
- Support for ACE5-Lite atomic transactions in the ACE-Lite TBU
- Support for LTI
- Support for a dedicated *Generic Interrupt Controller* (GIC) integration, with *Message Signaled Interrupts* (MSIs) supported for common interrupt types

Support for flexible integration

- You can place a configurable number of TBUs close to the masters being translated
- Communication between TBU and TCU over AXI4-Stream is supported using the supplied DTI interconnect components, or any other AXI4-Stream interconnect
- DTI interconnect components support hierarchical topologies and control the tradeoff between the number of wires and the DTI bandwidth

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Support for high-performance translation

- Scalable configurable MicroTLB and *Main TLB* (MTLB) in the TBU can reduce the number of translation requests to the TCU
- TBU direct indexing and MTLB partitioning enable the use of MTLB entries to be managed outside the TBU, improving real-time translation performance
- Optimization enables storage of all architecturally-defined page and block sizes, including contiguous page and block entries, as a single entry in the TBU and TCU TLBs (WCs)
- Per-TBU prioritization in the TCU enables high-priority transaction streams to be translated before low-priority streams
- TCU prefetch of translation tables, which can be enabled on a per-context basis, improves translation performance for real-time masters that access memory linearly
- *Hit-Under-Miss* (HUM) support in the TBU enables transactions with different AXI IDs to be propagated out of order, when a translation is available
- TBU detects multiple transactions that require the same translation so that only one TBU request to the TCU is required
- TCU detects multiple translations that require the same table in memory so that only one TCU memory request is required
- Multi-level, multi-stage walk caches in the TCU reduce translation cost by performing only part of the table walk process on a miss
- A configurable number of concurrent translations in the TBU and TCU promotes high translation throughput

Trace debugging

Using a CoreSight ELA-600 Embedded Logic Analyzer

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.4 Interfaces

Both the TCU and TBU support the following common interfaces:

- Clocks and resets
- *Distributed Translation Interface* (DTI)
- Tie-offs
- Interrupts
- PMU snapshot
- Test and debug
- LPI clock gating
- LPI powerdown

The TCU also supports the following interfaces:

- Programming
- System coherency
- *Queue and Table Walk* (QTW)/DVM
- *Generic Interrupt Controller* (GIC) *Message Signaled Interrupt* (MSI) interface

The ACE-Lite TBU also supports the following interfaces:

- *Transaction slave* (TBS)
- *Transaction master* (TBM)

The LTI TBU also supports the *Local Translation Interface* (LTI).

Related concepts

2.2 Interfaces on page 2-31

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.5 Configurable options

The MMU-700 is highly configurable and provides configuration options for each of the main components.

For the TCU, you can configure the following:

- Size of each cache
- Data width of the QTW/DVM interface
- Number of translations that can be performed at the same time
- Number of translation requests that can be accepted from all DTI masters

For the TBU, you can configure the following:

- Size of each cache
- Number of transactions that can be translated at the same time
- Register slices

For the ACE-Lite TBU, you can configure the following:

- Write data buffer depth
- Number of outstanding read and write transactions that the TBM interface supports
- Width of data, ID, User, StreamID, and SubstreamID signals on the TBS and TBM interfaces

Note

Depths are specified as a discrete number of entries.

You can also configure the DTI interconnect components to meet your system requirements.

See *2.5 Configuration options and methodology* on page 2-75.

Related concepts

2.5 Configuration options and methodology on page 2-75

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.6 Product documentation and design flow

This section describes the MMU-700 documentation in relation to the design flow.

1.6.1 Documentation

The MMU-700 documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the MMU-700. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that are described in the TRM are not relevant. If you are programming the MMU-700, then contact:

- The implementer to determine:
  - The build configuration of the implementation
  - The integration, if any, that was performed before implementing the MMU-700
- The integrator to determine the pin configuration of the device that you are using.

Configuration and Integration Manual

The *Configuration and Integration Manual* (CIM) describes:

- The available build configuration options and related issues in selecting them.
- How to integrate the MMU-700 into an SoC. This section describes the pins that the integrator must tie off to configure the macrocells for the required integration.
- The processes to sign off on the configuration, integration, and implementation of the design.

The CIM is a confidential book that is only available to licensees.

1.6.2 Design flow

The MMU-700 is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following processes:

Implementation

The implementer configures and synthesizes the RTL to produce a hard macrocell. This process might include integrating RAMs into the design.

Integration

The integrator connects the implemented design into an SoC. Integration includes connecting the design to a memory system and peripherals.

Programming

The system programmer develops the software to configure and initialize the MMU-700, and tests the required application software.

Each process is separate, and can include implementation and integration choices that affect the behavior and features of the MMU-700.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the following:

- Area
- Maximum frequency
- Features of the resulting macrocell

Configuration inputs

The integrator configures some features of the MMU-700 by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Software configuration

The programmer configures the MMU-700 by programming particular values into registers. This configuration affects the behavior of the MMU-700.

Related concepts

1.5 Configurable options on page 1-18

2.5 Configuration options and methodology on page 2-75

Related references

1.2 Compliance on page 1-13

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

1.7 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

r0p0-r0p1 The following changes apply to this release:

- New system discovery registers. See *3.9 TCU system discovery registers* on page 3-120 and *3.16 TBU system discovery registers* on page 3-153.
- New parameters. See *2.5.2 TCU buffer configuration options* on page 2-75 and *2.5.8 TBU buffer configuration options* on page 2-81.
- New stitching flow.
- New generate executable.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Chapter 2
Functional description

This chapter describes the functionality of the MMU-700.

It contains the following sections:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.1 About the functions

The major functional blocks of the MMU-700 are the *Translation Buffer Unit* (TBU), *Translation Control Unit* (TCU), and *Distributed Translation Interface* (DTI) interconnect.

The following figure shows an example system that uses the MMU-700.

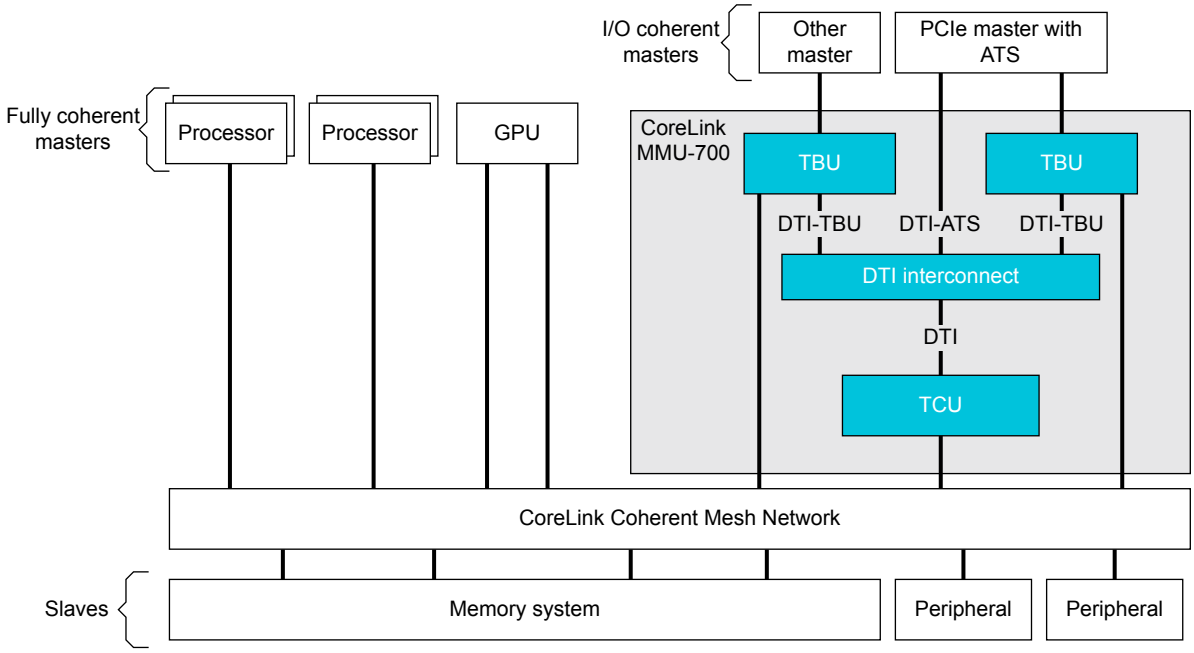

Figure 2-1 Example system with the MMU-700

The following figure shows an example system that uses the MMU-700 and includes a *Local Translation Interface* (LTI) TBU.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

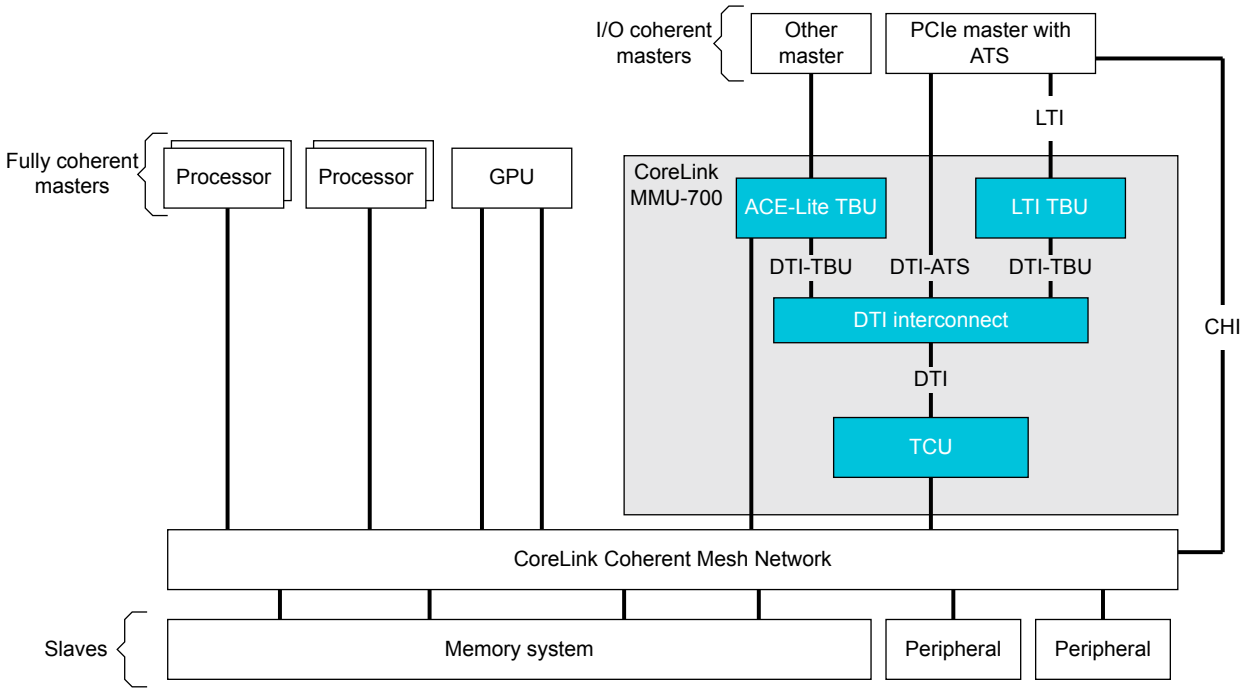

Figure 2-2 Example system with the MMU-700 and LTI TBU

The MMU-700 contains the following key components:

***Translation Buffer Unit* (TBU)**
The TBU contains *Translation Lookaside Buffers* (TLBs) that cache translation tables. The MMU-700 implements a TBU that can be connected to single master or multiple masters. It is also possible to connect multiple TBUs to a single master to improve performance. These TBUs are local to the corresponding master and can be one of the following:

- ACE-Lite TBU
- LTI TBU

***Translation Control Unit* (TCU)**
The TCU controls and manages the address translations. The MMU-700 implements a single TCU. In MMU-700-based systems, the AMBA DTI protocol defines the standard for communicating with the TCU. See the *AMBA® DTI Protocol Specification*.

DTI interconnect
The DTI interconnect connects multiple TBUs to the TCU.

When an MMU-700 TBU receives a transaction on the TBS or LA interface, it looks for a matching translation in its TLBs. If it has a matching translation, it uses it to translate the transaction and outputs the transaction on the TBM interface. If it does not have a matching translation, it requests a new translation from the TCU using the DTI interface.

When the TCU receives a DTI translation request, it uses the QTW interface to perform:

- Configuration table walks, which return configuration information for the translation context
- Translation table walks, that return translation information that is specific to the transaction address

The TCU contains caches that reduce the number of configuration and translation table walks that are to be performed. Sometimes no walks are required.

When the TBU receives the translation from the TCU, it stores it in its TLBs. If the translation was successful, the TBU uses it to translate the transaction, otherwise it terminates it.

A processor controls the TCU by:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- Writing commands to a Command queue in memory
- Receiving events from an Event queue in memory
- Writing to its configuration registers using the programming interface

See the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* for more information about the following:

- Translation
- How software communicates with the TCU

This section contains the following subsections:

- *2.1.1 Translation Buffer Unit* on page 2-25.
- *2.1.2 Translation Control Unit* on page 2-27.
- *2.1.3 DTI interconnect* on page 2-30.

2.1.1 Translation Buffer Unit

A typical SMMUv3-based system includes multiple *Translation Buffer Units* (TBUs). Each TBU is located close to the component for which it provides address translation.

A TBU can be one of the following:

- ACE-Lite TBU
- *Local Translation Interface* (LTI) TBU

A TBU intercepts transactions and provides the required translation from a *Translation Lookaside Buffer* (TLB) if possible. If a TLB does not contain the required translation, the TBU requests translations from the TCU and then caches the translation in one of the TLBs.

The following figure shows the ACE-Lite TBU.

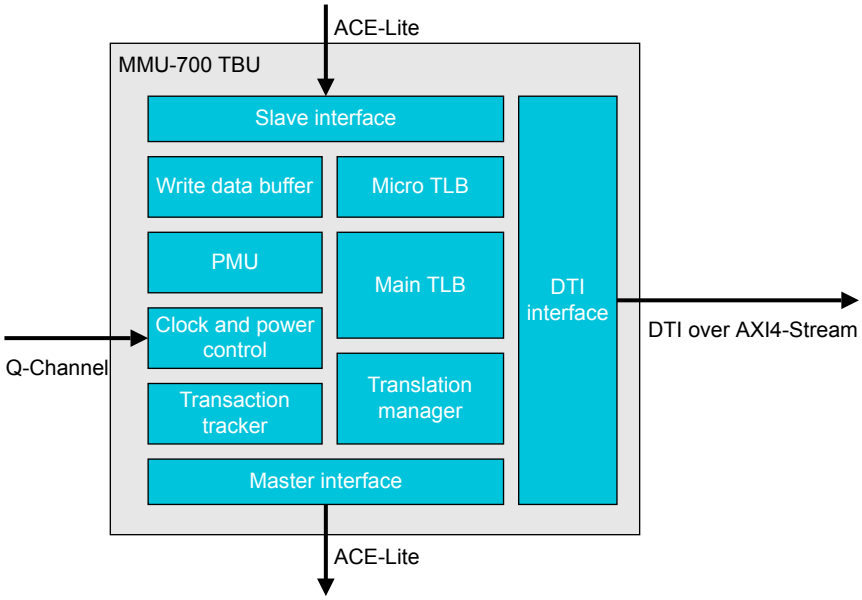

Figure 2-3 MMU-700 ACE-Lite TBU

The following figure shows the LTI TBU.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

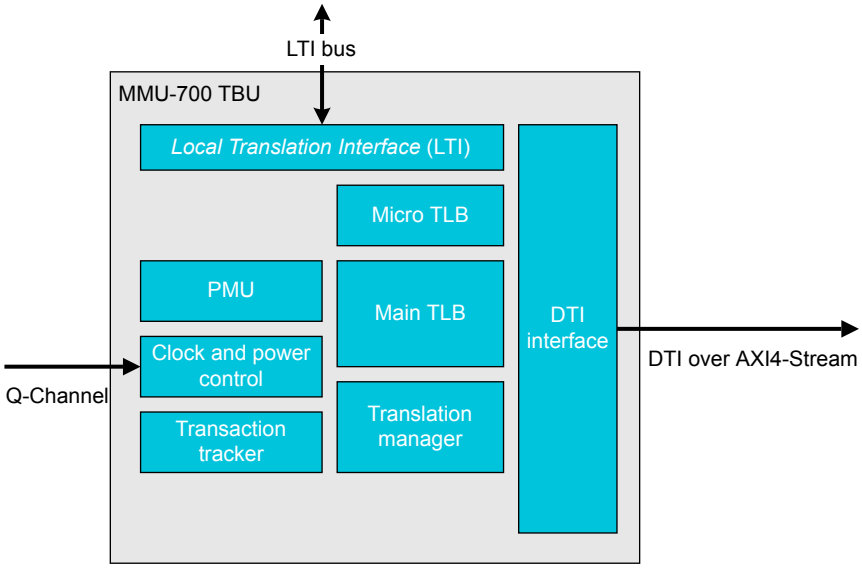

Figure 2-4 MMU-700 LTI TBU

The TBU consists of:

Master and slave interfaces

ACE-Lite TBU
For the TBS and TBM interfaces.

LTI TBU
For the LTI.

MicroTLB
The TBU compares incoming transactions with translations that are cached in the micro TLB before looking in the *Main TLB* (MTLB). The MicroTLB provides end-to-end translation from an input address to an output address. You can use a tie-off signal to configure the cache replacement policy as either round-robin or *Pseudo Least Recently Used* (PLRU).

Main TLB
Each TBU includes an optional *Main TLB* (MTLB) that caches translation table walk entries from:

- Stage 1 translations
- Stage 2 translations
- Stage 1 combined with stage 2 translations

The MTLB is a set associative cache structure with a configurable number of ways and banks.

If multiple translation sizes are in use, a single transaction might require multiple lookups. Lookups are pipelined to permit a sustained rate of one lookup per cycle.

TBU direct indexing enables the MMU-700 to manage MTLB entries externally to the TBU. Direct indexing improves the predictability of TBU performance, for bus masters that have real-time performance requirements.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

TBU hazarding

If the MicroTLB lookup results in a miss, the transaction checks whether there are any pending transactions from which it can use the translation. This method is called *forming a hazard*. If the transaction hazards on any pending transactions, then the transaction waits until the response is available for the hazarded transaction and uses that response. The number of unique addresses that form a hazard is limited. The `TBUCFG_HZRD_ENTRIES` parameter controls the number of unique addresses. For information about `TBUCFG_HZRD_ENTRIES`, see *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78 and *2.5.6 LTI TBU configuration options* on page 2-80.

For more information about TBU hazarding, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

Translation manager

The translation manager manages translation requests that are in progress. Each transaction occupies a translation slot until it is propagated downstream through the ACE-Lite TBM, or an LTI translation response is returned. All transactions are hazard-checked to reduce the possibility of duplicate translation requests being sent to the TCU.

There is no restriction on the ordering of transactions with different AXI IDs/LTI *Order Groups* (OGs). Transactions with different AXI IDs can be propagated downstream out-of-order.

All transactions with a given AXI ID/LTI OG value must remain ordered. The translation manager propagates such transactions when the translation is ready, provided no other transaction with the same AXI ID/LTI OG is already waiting.

For more information about AXI transaction identifiers, see the *AMBA® AXI and ACE Protocol Specification*.

For more information about LTI OGs, see the *AMBA® LTI Protocol Specification*.

Write data buffer

The write data buffer is available in the ACE-Lite TBU only. The optional write data buffer enables write transactions with different AXI IDs to progress through the TBU out-of-order. It reorders the data to match the downstream transaction order.

PMU

The PMU counts TBU performance-related events.

Clock and power control

The TBU has its own clock and power control, that the Q-Channels provide.

DTI interface

The master DTI interface uses the DTI protocol, typically over AXI4-Stream, to enable the TBU to communicate with a slave component. For the MMU-700, the slave component is the TCU. Although you can implement DTI over different transport protocols, the MMU-700 interfaces use AXI4-Stream.

Transaction tracker

The transaction trackers manage outstanding read and write transactions, permitting invalidation and synchronization to take place without stalling the AXI interfaces.

Related concepts

2.3 Operation on page 2-40

Related references

2.3.3 TBU direct indexing and MTLB partitioning on page 2-46
3.2 SMMU architectural registers on page 3-88

2.1.2 Translation Control Unit

A typical SMMUv3-based system includes a single *Translation Control Unit* (TCU). The TCU is usually the largest block in the system, and performs several roles.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The TCU:

- Manages the memory queues
- Performs translation table walks
- Performs configuration table walks
- Implements backup caching structures
- Implements the SMMU programmers model

The following figure shows the TCU with its main interfaces.

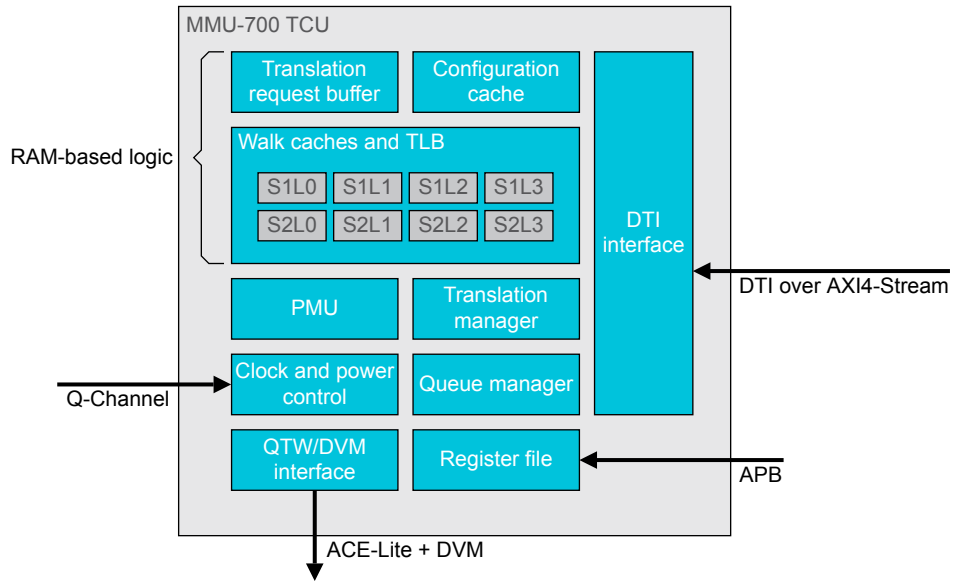

Figure 2-5 MMU-700 TCU

The TCU consists of:

Walk cache

The TCU is a set-associative walk cache that has a configurable number of banks and ways and stores the results of translation table walks. During MMU-700 configuration, the cache line entries can be programmatically split to create separate walk caches that are reserved for:

- Stage 1 level 0 table entries
- Stage 1 level 1 table and block entries
- Stage 1 level 2 table and block entries
- Stage 1 level 3 table entries
- Stage 2 level 0 table entries
- Stage 2 level 1 table and block entries
- Stage 2 level 2 table and block entries
- Stage 2 level 3 table entries

To enable and disable the walk cache for a particular stage and level of translation, use the *3.7.1 TCU_CTRL register* on page 3-106. If an error occurs for a cache line entry, the *3.8.3 TCU_ERRSTATUS register* on page 3-116 identifies the affected entry.

The walk cache is useful in cases where a translation request results in a miss in other TCU caches. A subsequent hit in the walk cache requires only a single memory access to complete the translation table walk and fetch the required descriptor.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Configuration cache

The configuration caches are 4-way set-associative cache structures that store configuration information. Each entry stores the *Context Descriptor* (CD) and *Stream Table Entry* (STE) contents for a translation context.

Note

The configuration cache does not cache the contents of intermediate configuration tables.

Translation manager

The translation manager manages translation requests that are in progress. All translation table walks and configuration table walks are hazard-checked to reduce the possibility of multiple transactions requesting duplicate walks.

Translation request buffer

The translation request buffer stores translation requests from TBUs when all translation manager slots are full. The translation request buffer supports more slots than the translation manager. When correctly configured, this buffer has enough space to store all translation requests that TBUs can issue simultaneously. This buffer therefore prevents the DTI interface from becoming blocked.

PMU

The PMU counts TCU performance-related events and has a configurable number of counters to count the events.

Clock and power control

The TCU has its own clock and power control, that the Q-Channels provide.

Queue manager

The queue manager manages all SMMUv3 Command queues and Event queues that are stored in memory.

QTW/DVM interface

The *Queue and Table Walk* (QTW)/*Distributed Virtual Memory* (DVM) interface is an ACE-Lite +DVM master interface.

Register file

The register file implements the SMMUv3 programmers model, as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

DTI interface

The slave DTI interface uses the DTI protocol, typically over AXI4-Stream, to enable the TCU to communicate with a master component. For the MMU-700, the master component is either a TBU or a PCIe master.

Related concepts

2.2 Interfaces on page 2-31
2.3.7 TCU transaction handling on page 2-51
2.3.8 TCU prefetch on page 2-51
2.3 Operation on page 2-40

Related references

3.2 SMMU architectural registers on page 3-88

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.1.3 DTI interconnect

The TCU and TBUs use a DTI interface to communicate. The DTI interconnect enables the DTI interface to use the AXI4-Stream transport protocol.

The DTI interconnect can connect any components that conform to the AXI4-Stream protocol, as the *Arm® AMBA® Distributed Translation Interface (DTI) Protocol Specification* defines.

The DTI interconnect contains internal components that are hierarchically composable, that is, they can be connected in different ways to suit your system requirements. For example, within an MMU-700 system, you can use the switch component to combine the DTI interfaces of multiple TBUs into a single DTI interface. You can then connect the combined DTI interface to another DTI interconnect that is closer to the TCU.

The DTI interconnect includes switch, sizer, and register slice components.

Switch

The switch connects multiple DTI masters, such as TBUs, to a DTI slave such as a TCU. The switch implements the following parallel networks:

- For TBU to TCU traffic, a network that connects multiple AXI4-Stream slave interfaces to a single AXI4-Stream master interface
- For TCU to TBU traffic, a network that connects a single AXI4-Stream slave interface to multiple AXI4-Stream master interfaces

——— **Note** ———

The switch does not store any data, and therefore does not require a Q-Channel clock gating interface.

———————————

Sizer

The sizer connects channels that have different data widths, enabling different tradeoffs of bandwidth to area. The sizer supports conversion between any of the supported AXI4-Stream data widths:

- 1 byte
- 4 bytes
- 10 bytes
- 20 bytes

The sizer includes a Q-Channel interface to provide clock gating control.

Register slice

Use the register slice to improve timing. The register slice includes a Q-Channel interface to provide clock gating control.
The MMU-700 DTI interconnect components do not include a component to connect different clock and power domains. You can connect DTI interfaces in different clock and power domains by using the *Bidirectional AXI4-Stream* (BAS) configuration of the ADB-400 AMBA Domain Bridge.

Related concepts

2.3 Operation on page 2-40

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.2 Interfaces

The MMU-700 includes interfaces for each of the TCU, TBU, and DTI interconnect components.

The DTI interconnect consists of switch, sizer, and register slice components that you can connect separately, and these components therefore have their own interfaces.

The PMU snapshot interface is common to both TCU and TBU.

This section contains the following subsections:

- *2.2.1 TCU interfaces* on page 2-31.
- *2.2.2 TBU interfaces* on page 2-34.
- *2.2.3 DTI interconnect interfaces* on page 2-37.

2.2.1 TCU interfaces

The MMU-700 TCU includes several master and slave interfaces.

The following figure shows the TCU interfaces.

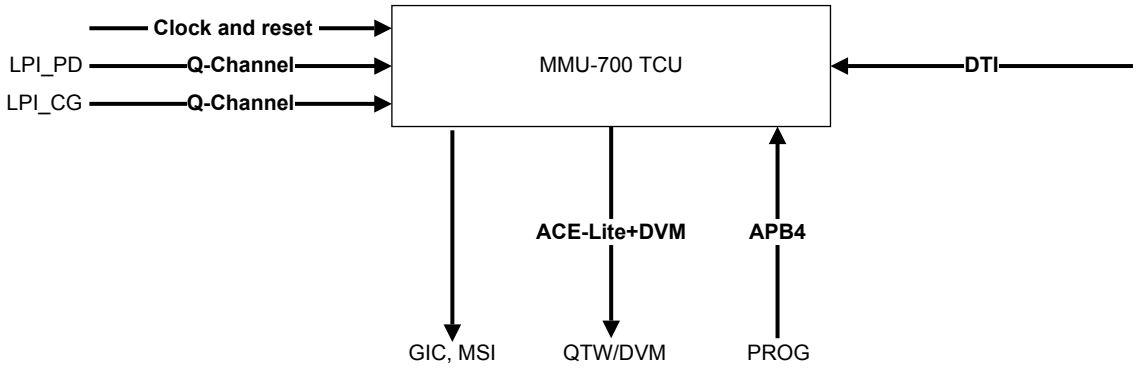

Figure 2-6 TCU interfaces

TCU Queue and Table Walk/Distributed Virtual Memory interface

The *Queue and Table Walk/Distributed Virtual Memory* (QTW/DVM) interface is an ACE-Lite+DVM master interface.

The QTW/DVM interface issues the following transaction types:

- ReadNoSnoop
- WriteNoSnoop
- ReadOnce
- WriteUnique
- DVM Complete

The QTW/DVM interface uses the write address transaction ID signal **awid_qtw**, and the read address transaction ID signal, **arid_qtw**.

External ID Width = `TCU_ID_WIDTH` = MAX(4, ceil($\log_2$(`TCUCFG_PTW_SLOTS`)) + 2).

The smallest possible `TCU_ID_WIDTH` value is 4.

See *2.5 Configuration options and methodology* on page 2-75.

The following table shows the possible values of **arid_qtw**.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-1 arid_qtw assignment

Transaction type	arid_qtw[1:0]	arid_qtw[TCU_ID_WIDTH-1:2]
Command Queue walk	2'b00	Bits [3:2] = 2'b00. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH - 1 :4} are 0.
DVM Complete	2'b00	Bits [3:2] = 2'b01. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH - 1 :4} are 0.
Configuration table walk	2'b01	Indicates the configuration table walk slot that is requesting the configuration table walk
Page table walk	2'b10	Indicates the page table walk slot that is requesting the page table walk

The following table shows the possible values of **arid_qtw**.

Table 2-2 awid_qtw assignment

Transaction type	awid_qtw[1:0]	awid_qtw[TCU_ID_WIDTH-1:2]
PRI Queue Write	2'b00	Bits [3:2] = 2'b01. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH - 1:4} are 0.
Event Queue write	2'b00	Bits [3:2] = 2'b10. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH - 1:4} are 0.
MSI write	2'b00	Bits [3:2] = 2'b11. If TCU_ID_WIDTH > 4, bits {TCU_ID_WIDTH - 1:4} are 0.
HTTU Write	2'b11	Indicates the page table walk slot requesting the HTTU write

To support 16-bit *Virtual Machine IDentifiers* (VMIDs), the interface provides DVMv8.4 support.

The interface does not issue cache maintenance operations or exclusive accesses.

TCU PROG interface

The PROG interface is an AMBA APB4 slave interface. It enables software to program the MMU-700 internal registers and read the *Performance Monitoring Unit* (PMU) registers and the Debug registers.

This interface runs synchronously with the other TCU interfaces.

The applicable address width for this interface depends on the value of TCUCFG_NUM_TBU:

- When TCUCFG_NUM_TBU = 14, the address width is 21 bits
- When TCUCFG_NUM_TBU = 62, the address width is 23 bits

Transactions are *Read-As-Zero, Writes Ignored* (RAZ/WI) when any of the following apply:

- An unimplemented register is accessed
- **PSTRB[3:0]** is not 0b1111 for write transfers
- **PPROT[1]** is not set to 0 for Secure register accesses

For more information, see the *AMBA® APB Protocol Specification*.

Related references

A.1.3 TCU programming interface signals on page Appx-A-173

TCU LPI_PD interface

This Q-Channel slave interface manages LPI powerdown for the TCU.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

For more information, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

Related references

A.1.6 TCU LPI_PD interface signals on page Appx-A-174

TCU LPI_CG interface

This Q-Channel slave interface enables LPI clock gating for the TCU.

For more information, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

Related references

A.1.7 TCU LPI_CG interface signals on page Appx-A-174

TCU DTI interface

The DTI interface manages communication between the TBUs and the TCU, using the DTI protocol. The DTI protocol can be conveyed over different transport layer mediums, including AXI4-Stream.

The TCU includes a slave DTI interface and each TBU includes a master DTI interface. To permit bidirectional communication, each DTI interface includes one AXI4-Stream master interface and one AXI4-Stream slave interface.

For more information, see the *AMBA® DTI Protocol Specification* and the *AMBA® 4 AXI4-Stream Protocol Specification*.

Related concepts

2.3.1 DTI overview on page 2-40

Related references

A.1.8 TCU DTI interface signals on page Appx-A-175

TCU interrupt interfaces

This interface provides global, per-context, and performance interrupts. A direct MSI interface to a *Generic Interrupt Controller* (GIC) is also supported, to avoid complex dependencies in the system.

Related references

A.1.10 TCU MSI interface signals on page Appx-A-176

TCU SYSCO signaling

The MMU-700 provides a hardware system coherency interface. This master interface permits the TCU to remove itself from a coherency domain in response to an LPI request.

The SYSCO signals include the **syscoreq_qtw** and **syscoack_qtw** handshake signals to enter or exit a coherency domain.

If the **sup_btm** signal is tied LOW, the **syscoreq_qtw** signal is always driven LOW and **syscoack_qtw** is ignored.

Related references

A.1.13 TCU ELA debug signals on page Appx-A-180

TCU tie-off signals

The TCU tie-off signals enable you to initialize various operating parameters on exit from reset state.

Related references

A.1.12 TCU tie-off signals on page Appx-A-179

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

TCU ELA observation interface

This *Embedded Logic Analyzer* (ELA) observation master interface drives the signal group, signal qualifier, and signal clock enable ELA signals to the on-chip ELA module, if present.

When `TCUCFG_USE_ELA_DEBUG` is 0, these signals are tied to 0. See *2.5.3 TCU debug configuration options* on page 2-78.

For more information about the interface signals, see *B.1 TCU observation interfaces* on page Appx-B-204.

Related concepts

2.3.6 Distributed Virtual Memory (DVM) messages on page 2-50
2.3.9 Error responses on page 2-53
2.4.2 AMBA implementation on page 2-60

Related references

A.1.2 TCU QTW/DVM interface signals on page Appx-A-170

2.2.2 TBU interfaces

Each MMU-700 TBU includes several master and slave interfaces.

The following figure shows the ACE-Lite TBU interfaces.

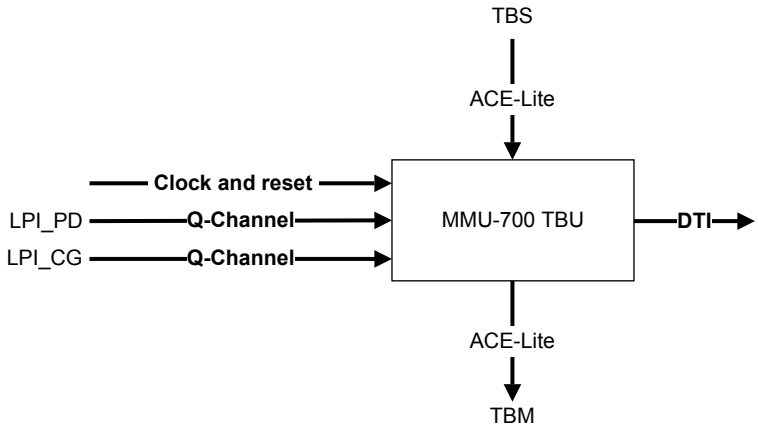

Figure 2-7 ACE-Lite TBU interfaces

The following figure shows the LTI TBU interfaces.

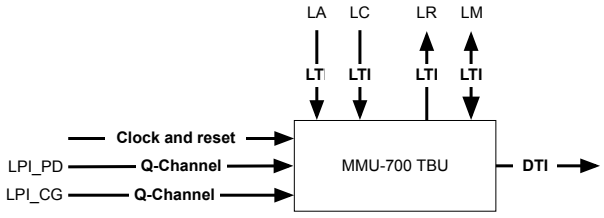

Figure 2-8 LTI TBU interfaces

ACE-Lite TBU TBS interface

The transaction slave interface, TBS, is an ACE5-Lite interface on which the ACE-Lite TBU receives incoming untranslated memory accesses.

This interface supports a 64-bit address width.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The interface implements optional signals to support the following AXI5 extensions:

- Wakeup_Signals
- Untranslated_Transactionsv2
- Cache_Stash_Transactions
- DeAllocation_Transactions
- Atomic_Transactions
- Loopback_Signals
- Poison
- Unique_ID_Support
- Read_Data_Chunking
- CMO_On_Read, Persist_CMO

For more information, see *2.4.2 AMBA implementation* on page 2-60.

The TBS interface supports ACE Exclusive accesses.

If a transaction is terminated in the TBU, the transaction tracker returns the transaction with the user-defined AXI **RUSER** and **BUSER** bits set to 0.

Related concepts

2.3.9 Error responses on page 2-53

Related references

A.2.2 TBU TBS interface signals on page Appx-A-183

ACE-Lite TBU TBM interface

The transaction master interface, TBM, is an ACE5-Lite interface on which the ACE-Lite TBU sends outgoing translated memory accesses.

The AXI ID of a transaction on this interface is the same as the AXI ID of the corresponding transaction on the TBS interface.

This interface supports a 52-bit address width, and `TBUCFG_DATA_WIDTH` defines the data width. See:

- *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78
- *2.5.6 LTI TBU configuration options* on page 2-80

This interface can issue read and write transactions until the outstanding transaction limit is reached. The MMU-700 provides parameters that permit you to configure:

- The outstanding read transactions limit
- The outstanding write transactions limit
- The total outstanding read and write transactions limit

The interface implements optional signals to support the following AXI5 extensions:

- Wakeup_Signals
- Untranslated_Transactionsv2[a]
- Cache_Stash_Transactions
- DeAllocation_Transactions
- Atomic_Transactions
- Loopback_Signals
- Ordered Write Observation
- Poison
- Unique_ID_Support
- Read_Data_Chunking
- Read_Interleaving_Disabled[b]
- CMO_On_Read, Persist_CMO
- MPAM_Support

[a] The TBM does not support the *Untranslated_Transactions* property. The TBM contains the **axmmusecsid** and **axmmusid** signals for backwards-compatability with other SMMU products. These signals are not required for normal operation of the MMU-700 and you can ignore them.

[b] The BIU supports the *Read_Interleaving_Disabled* property provided that terminated transaction responses are not interleaved.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

For more information, see 2.4.2 AMBA implementation on page 2-60.

When receiving an SLVERR or DECERR response to a downstream transaction, the TBM interface propagates the same response to the TBS interface.

Related concepts

2.3.9 Error responses on page 2-53
2.4.2 AMBA implementation on page 2-60

Related references

A.2.3 TBU TBM interface signals on page Appx-A-187

LTI TBU LTI interface

The LTI interface supports two virtual channels, one for reads and one for writes. User signals are not implemented. You can define other LTI properties by using configuration parameters.

The interface contains the following channels:

LA Request channel. Address and attributes that require translation are sent to the TBU.

LR Response channel. Provides the translated address and attributes to the LTI device.

LC Completion channel. LTI devices must provide information about completion to the TBU.

LM Link Management channel. Contains:

- **LMOPENREQ**
- **LMOPENACK**
- **LMASKCLOSE**
- **LMACTIVE**

For more information, see the following:

- 2.4.4 LTI implementation on page 2-74
- 2.5 Configuration options and methodology on page 2-75
- *AMBA® LTI Protocol Specification*

TBU LPI_PD interface

This Q-Channel slave interface manages LPI powerdown for the TBU.

For more information, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

Related references

A.2.5 TBU LPI_PD interface signals on page Appx-A-190

TBU LPI_CG interface

This Q-Channel slave interface enables LPI clock gating for the TBU.

For more information, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

Related references

A.2.6 TBU LPI_CG interface signals on page Appx-A-191

TBU DTI interface

The TBU DTI interface enables master devices with their own TLB and prefetch capability to request translations from the MMU-700. This interface uses the DTI-TBU protocol for communication between the TBU and the TCU.

The TCU includes a slave DTI interface and each TBU includes a master DTI interface. To permit bidirectional communication, each DTI interface includes one AXI4-Stream master interface and one AXI4-Stream slave interface.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

For more information, see the *AMBA® DTI Protocol Specification* and the *AMBA® 4 AXI4-Stream Protocol Specification*.

Related concepts

2.3.1 DTI overview on page 2-40

Related references

A.2.7 TBU DTI interface signals on page Appx-A-191

TBU interrupt interfaces

This interface provides global, per-context, and performance interrupts.

Related references

A.2.8 TBU interrupt signals on page Appx-A-192

TBU tie-off signals

The TBU tie-off signals enable you to initialize various operating parameters on exit from reset state.

Related references

A.2.9 TBU tie-off signals on page Appx-A-192

TBU ELA observation interface

This *Embedded Logic Analyzer* (ELA) observation master interface drives the signal group, signal qualifier, and signal clock enable ELA signals to the on-chip ELA module, if present.

When `TBUCFG_USE_ELA_DEBUG` is 0, these signals are tied to 0. See *2.5.9 TBU debug configuration options* on page 2-82.

For more information about the interface signals, see:

- *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209
- *B.3 LTI TBU observation interfaces* on page Appx-B-213

2.2.3 DTI interconnect interfaces

The DTI interconnect includes interfaces for each of the switch, sizer, and register slice components.

DTI interconnect switch interfaces

The DTI interconnect switch component includes dedicated interfaces.

The following figure shows the DTI interconnect switch interfaces.

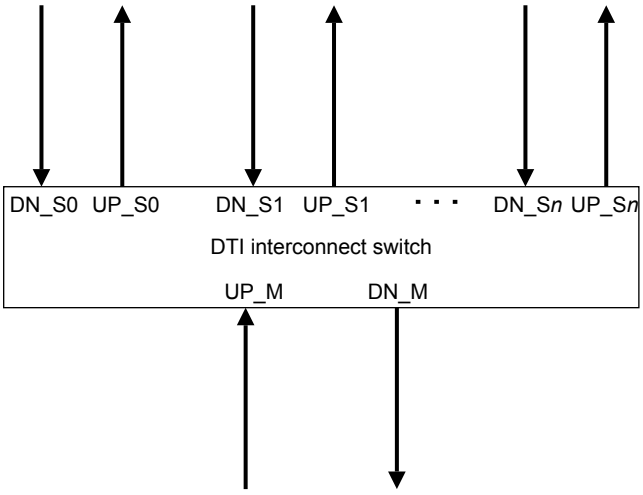

Figure 2-9 DTI interconnect switch interfaces

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The following table provides more information about the switch interfaces.

Table 2-3 DTI interconnect switch interfaces

Interface	Interface type	Protocol	Description
DN_Sn	Slave	AXI4-Stream	Slave downstream interface. One DN_Sn interface is present for each slave interface.
UP_Sn	Master		Slave upstream interface. One UP_Sn interface is present for each slave interface.
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

Note

The interconnect switch does not store any data, and therefore does not require a Q-Channel clock-gating interface.

DTI interconnect sizer interfaces

The DTI interconnect sizer component includes dedicated interfaces.

The following figure shows the DTI interconnect sizer interfaces.

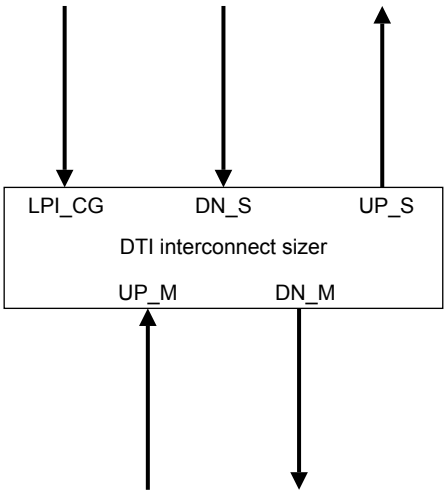

Figure 2-10 DTI interconnect sizer interfaces

The following table provides more information about the sizer interfaces.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-4 DTI interconnect sizer interfaces

Interface	Interface type	Protocol	Description
LPI_CG	Slave	Q-Channel	Clock gating interface
DN_S	Slave	AXI4-Stream	Slave downstream interface
UP_S	Master		Slave upstream interface
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

DTI interconnect register slice interfaces

The DTI interconnect register slice component includes dedicated interfaces.

The following figure shows the DTI interconnect register slice interfaces.

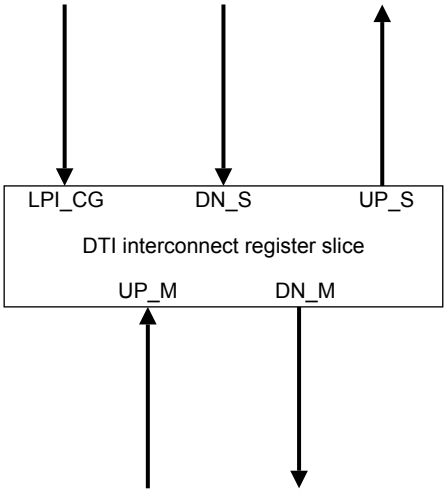

Figure 2-11 DTI interconnect register slice interfaces

The following table provides more information about the register slice interfaces.

Table 2-5 DTI interconnect register slice interfaces

Interface	Interface type	Protocol	Description
LPI_CG	Slave	Q-Channel	Clock gating interface
DN_S	Slave	AXI4-Stream	Slave downstream interface
UP_S	Master		Slave upstream interface
DN_M	Master		Master downstream interface
UP_M	Slave		Master upstream interface

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.3 Operation

This section provides information about the operation of the MMU-700 features.

This section contains the following subsections:

2.3.1 DTI overview

In an MMU-700-based system, the AMBA DTI protocol defines the standard for communicating with a TCU.

The AMBA DTI protocol includes both:

- DTI-TBU protocol, for communication between a TBU and a TCU
- DTI-ATS protocol, for communication between a PCIe Root Complex and a TCU

The DTI protocol is a point-to-point protocol. Each channel consists of a link, a DTI master, and a DTI slave. The DTI masters in the respective protocols are:

- The TBU, in the DTI-TBU protocol
- The PCIe Root Complex, in the DTI-ATS protocol

The DTI slave in both DTI-TBU and DTI-ATS is the TCU.

DTI masters and slaves communicate using defined DTI messages. The DTI protocol defines the following message groups:

- Page request
- Register access
- Translation request
- Connection and disconnection
- Invalidation and synchronization

The DTI_TBU_CONDIS_REQ message initiates a TBU connection or disconnection handshake. The TBU uses this message to connect to the TCU. During connection, the TBU can specify the number of requested translation tokens. The DTI master uses the TOK_TRANS_REQ field to request translation tokens. For the TBU, the **max_tok_trans** signal defines the number of translation tokens that the TBU requests.

The TBU uses the TOK_INV_GNT field to grant invalidation tokens. The TBU grants only one invalidation token, and the TCU is only capable of issuing one invalidate message at a time.

A DTI master uses a DTI_TBU_CONDIS_REQ or a DTI_ATS_CONDIS_REQ message to initiate a connection handshake. If the master provides a **TID** value that is greater than the maximum supported **TID** that `TCUCFG_NUM_TBU` defines, the slave sends a Connect Deny message.

A translation request to the TCU where StreamID $\geq 2^{24}$ results in a fault and an SMMUv3 C_BAD_STREAMID event. If the TBU receives an invalidation request where StreamID $\geq 2^{24}$, any

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

comparisons with a StreamID value fail. No TLB entries are invalidated, but other effects that do not consider the supplied StreamID occur as normal.

Note

- The TBU never generates translation requests with StreamID $\geq 2^{24}$
- The TCU never generates invalidation requests with StreamID $\geq 2^{24}$

For more information, see the *AMBA® DTI Protocol Specification*.

2.3.2 Performance Monitoring Unit

The MMU-700 includes a PMU for the TCU and a PMU for each TBU. The PMU events and counters indicate the runtime performance of the MMU-700.

The MMU-700 includes logic to gather various statistics on the operation of the MMU during runtime, using events and counters. These events, which the SMMUv3 architecture defines, provide useful information about the behavior of the MMU. You can use this information when debugging or profiling traffic.

SMMUv3 architectural performance events

Both the TCU and the TBU implement performance events that the SMMUv3 Performance Monitor extension defines.

The SMMU_PMCG_SMR0 register can filter some events so that only events with a particular StreamID are counted. This event filtering includes:

- Speculative transactions and translations
- Transactions and translations that result in a terminated transaction or a translation fault

The following table shows the architecturally defined MMU-700 TCU performance events.

Table 2-6 SMMUv3 performance events for the TCU

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
Clock cycle	0x0	No	Counts clock cycles. Cycles where the clock is gated after a clock Q-Channel handshake are not counted.
Transaction	0x1	Yes	Counts translation requests that originate from a DTI-TBU or DTI-ATS master
TLB miss caused by incoming transaction or translation request	0x2	Yes	Counts translation requests where the translation walks new translation table entries
Configuration cache miss caused by transaction or translation request	0x3	Yes	Counts translation requests where the translation walks new configuration table entries
Translation table walk access	0x4	Yes	Counts translation table walk accesses
Configuration structure access	0x5	Yes	Counts configuration table walk accesses
PCIe ATS Translation Request received	0x6	Yes	Counts translation requests that originate from a DTI-ATS master

The following table shows the architecturally defined MMU-700 TBU performance events.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-7 SMMUv3 performance events for the TBU

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
Clock cycle	0x0	No	Counts clock cycles. Cycles where the clock is gated after a clock Q-Channel handshake are not counted.
Transaction	0x1	Yes	Counts transactions that are issued on the TBM interface
TLB miss caused by incoming transaction or translation request	0x2	Yes	Counts non-speculative translation requests that are issued to the TCU
PCIe ATS Translation Request received	0x7	Yes	Counts ATS-translated transactions that are issued on the TBM interface

For more information, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

MMU-700 TCU events

The MMU-700 PMU can be configured to monitor a range of IMPLEMENTATION DEFINED TCU performance events.

The SMMU_PMCG_SMR0 register can filter some TCU performance events so that only events with a particular StreamID are counted. This event filtering includes:

- Speculative transactions and translations
- Transactions and translations that result in a terminated transaction or a translation fault

The following table shows the TCU performance events.

Table 2-8 MMU-700 TCU performance events

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
S1L0WC lookup	0x80	Yes	Counts translation requests that access the S1L0WC walk cache
S1L0WC miss	0x81	Yes	Counts translation requests that access the S1L0WC walk cache and do not result in a hit
S1L1WC lookup	0x82	Yes	Counts translation requests that access the S1L1WC walk cache
S1L1WC miss	0x83	Yes	Counts translation requests that access the S1L1WC walk cache and do not result in a hit
S1L2WC lookup	0x84	Yes	Counts translation requests that access the S1L2WC walk cache
S1L2WC miss	0x85	Yes	Counts translation requests that access the S1L2WC walk cache and do not result in a hit
S1L3WC lookup	0x86	Yes	Counts translation requests that access the S1L3WC walk cache
S1L3WC miss	0x87	Yes	Counts translation requests that access the S1L3WC walk cache and do not result in a hit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-8 MMU-700 TCU performance events (continued)

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
S2L0WC lookup	0x88	Yes	Counts translation requests that access the S2L0WC walk cache
S2L0WC miss	0x89	Yes	Counts translation requests that access the S2L0WC walk cache and do not result in a hit
S2L1WC lookup	0x8A	Yes	Counts translation requests that access the S2L1WC walk cache
S2L1WC miss	0x8B	Yes	Counts translation requests that access the S2L1WC walk cache and do not result in a hit
S2L2WC lookup	0x8C	Yes	Counts translation requests that access the S2L2WC walk cache
S2L2WC miss	0x8D	Yes	Counts translation requests that access the S2L2WC walk cache and do not result in a hit
S2L3WC lookup	0x8E	Yes	Counts translation requests that access the S2L3WC walk cache
S2L3WC miss	0x8F	Yes	Counts translation requests that access the S2L3WC walk cache and do not result in a hit
WC read	0x90	Yes	Counts reads from the walk cache RAMs, excluding reads that invalidation requests cause **Note** A single walk cache lookup might result in multiple RAM reads. This behavior permits contiguous entries to be located.
Buffered translation	0x91	Yes	Counts translations written to the translation request buffer because all translation slots are full
CC lookup	0x92	Yes	Counts lookups into the configuration cache
CC read	0x93	Yes	Counts reads from the configuration cache RAMs, excluding reads that invalidation requests cause **Note** A single cache lookup might result in multiple RAM reads. This behavior permits contiguous entries to be located.
CC miss	0x94	Yes	Counts lookups into the configuration cache that result in a miss
Speculative translation	0xA0	Yes	Counts translation requests that are marked as Speculative

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Note

A single DTI translation request might correspond to multiple translation request events in either of the following circumstances:

- A translation results in a stall fault event and is restarted
- If a translation results in a stall fault event, because the Event queue is full, the translation is retried when an Event queue slot becomes available

MMU-700 TBU events

The MMU-700 PMU can be configured to monitor a range of IMPLEMENTATION DEFINED TBU performance events.

The SMMU_PMCG_SMR0 register can filter the TBU performance events so that only events with a particular StreamID are counted. This event filtering includes:

- Speculative transactions and translations
- Transactions and translations that result in a terminated transaction or a translation fault

The following table shows the TBU performance events.

Table 2-9 MMU-700 TBU performance events

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
Main TLB lookup	0x80	Yes	Counts Main TLB lookups
Main TLB miss	0x81	Yes	Counts translation requests that miss in the Main TLB
Main TLB read	0x82	Yes	Counts once per access to the Main TLB RAMs, excluding reads that invalidation requests cause **Note** A transaction might access the Main TLB multiple times to look for different page sizes.
Micro TLB lookup	0x83	Yes	Counts Micro TLB lookups
Micro TLB miss	0x84	Yes	Counts translation requests that miss in the Micro TLB
Slots full	0x85	No	Counts once per cycle when all slots are occupied and not ready to issue transactions downstream. This Secure event is visible only when the SMMU_PMCG_SCR.SO bit is set to 1.
Out of translation tokens	0x86	No	Counts once per cycle when a translation request cannot be issued because all translation tokens are in use. This Secure event is visible only when the SMMU_PMCG_SCR.SO bit is set to 1.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-9 MMU-700 TBU performance events (continued)

Event	Event ID	SMMU_PMCG_SMR0 filterable	Description
Write data buffer full	0x87	No	Counts once per cycle when a transaction is blocked because the write data buffer is full. This Secure event is visible only when the SMMU_PMCG_SCR.SO bit is set to 1.
DCMO downgrade	0x8B	Yes	For the ACE-Lite TBU, counts when either: • A MakeInvalid transaction on the TBS interface is output as CleanInvalid on the TBM interface • A ReadOnceMakeInvalid transaction on the TBS interface is output as ReadOnceCleanInvalid on the TBM interface For the LTI TBU, counts once per cycle when an LTI DCMO or R-DCMO transaction on the LA channel is responded to with a downgrade on the LR channel
Stash fail	0x8C	Yes	For the ACE-Lite TBU, counts when either: • A WriteUniquePtlStash or WriteUniqueFullStash transaction on TBS is output as a WriteNoSnoop or WriteUnique transaction on the TBM interface • A StashOnceShared or StashOnceUnique transaction on the TBS interface has a valid translation, but is terminated in the TBU For the LTI TBU, counts once whenever either an: • LTI WDCP transaction on the LA channel is downgraded as W on the LR channel. • LTI DCP transaction on the LA channel that is responded to as FaultRAZWI on the LR channel is counted. This can be because of: — Memory attributes or DCP, R, W, or X permission check failure in the TLBU — DTI fault response with Non-Abort The transaction that is responded to with FaultAbort because of DTI StreamDisable or GlobalDisable is not counted **Note** A StashOnceShared or StashOnceUnique transaction that is terminated because of a StreamDisable or GlobalDisable translation response does not cause this event to count

SMMUv3 PMU register architectural options

The SMMUv3 architecture defines the *Performance Monitor Counter Group* (PMCG) configuration register, SMMU_PMCG_CFGR. An MMU-700 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

The following table shows the SMMU_PMCG_CFGR register options that the MMU-700 TCU and TBU use.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-10 MMU-700 SMMU_PMCG_CFGR register architectural options

Field		Default value	Description for default value
SID_FILTER_TYPE		1	A single StreamID filter applies to all PMCG counters
CAPTURE		1	Capture of counter values into SVRn registers is supported
MSI		0	The counter group does not support *Message Signaled Interrupts* (MSIs)
RELOC_CTRS		1	The PMCG registers are relocated to page 1 of the PMU address map
SIZE		`0x31`	The counter group implements 32-bit counters
MPAM		0	*Memory System Resource Partitioning and Monitoring* (MPAM)
NCTR	NCTR for the TCU	`TCUCFG_PMU_COUNTERS` - 1	The counter group includes `TCUCFG_PMU_COUNTERS` counters. See *2.5.2 TCU buffer configuration options* on page 2-75.
	TCTR for the TBU	`TBUCFG_PMU_COUNTERS` - 1	The counter group includes `TBUCFG_PMU_COUNTERS` counters. See *2.5.8 TBU buffer configuration options* on page 2-81.

Related references

3.3 MMU-700 memory map on page 3-93

PMU snapshot interface

The *Performance Monitoring Unit* (PMU) snapshot interface is included on the TCU and on each TBU. You can use this asynchronous interface to initiate a PMU snapshot. A simultaneous snapshot of each counter register is created and copied to the respective SMMU_PMCG_SVRn register.

The PMU snapshot sequence is a 4-phase handshake. Both **pmusnapshot_req** and **pmusnapshot_ack** are LOW after reset. A snapshot occurs on the rising edge of **pmusnapshot_req**, and is equivalent to writing the value 1 to SMMU_PMCG_CAPR.CAPTURE.

The **pmusnapshot_req** signal is sampled using synchronizing registers. A register drives **pmusnapshot_ack** so that the connected component can sample the signal asynchronously.

Related concepts

2.3.4 RAS implementation on page 2-47

Related references

A.1.5 TCU PMU snapshot interface signals on page Appx-A-174

A.2.4 TBU PMU snapshot interface signals on page Appx-A-190

2.3.3 TBU direct indexing and MTLB partitioning

TBU direct indexing can help your system to meet real-time translation requirements by enabling the MMU-700 to manage *Main TLB* (MTLB) entries externally to the TBU.

Note

If you use the direct indexing and MTLB partitioning features, MPAM is not valid.

Direct indexing enables real-time translation requirements to be met, as follows:

- It can be guaranteed that different streams do not overwrite prefetched entries
- The MTLB can be partitioned into different sets of entries that different streams use

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

If you configure your system to not use direct indexing, you can select MTLB partitioning. MTLB partitioning has similar behavior, but only the most significant TLB index bits are provided, and the other bits are generated internally.

Direct indexing is enabled for a TBU when TBUCFG_DIRECT_IDX = 1.

When TBUCFG_DIRECT_IDX = 1, or when an MTLB is partitioned, the width of the **AxUSER** signals on the TBS interface is extended to convey the indexing information that is required for TBU direct indexing or MTLB partitioning.

Note

The table shows the extended bits in the order MSB first.

Table 2-11 Extended aruser_s and awuser_s bits for MTLB partitioning

Field name	Width	Description
mtlbidx	When direct indexing is enabled, the width of this field is $\log_2$(TBUCFG_MTLB_DEPTH) - $\log_2$(TBUCFG_MTLB_WAYS). When direct indexing is not enabled, the width of this field is 0.	MTLB index
mtlbway	When direct indexing is enabled, the width of this field is $\log_2$(TBUCFG_MTLB_WAYS). When direct indexing is not enabled, the width of this field is 0.	MTLB way
mtlbpart	$\log_2$(TBUCFG_MTLB_PARTS)	MTLB partition
-	TBUCFG_AWUSER_WIDTH for **awuser_s**. TBUCFG_ARUSER_WIDTH for **aruser_s**.	Regular **AxUSER** signals

If an MTLB is partitioned:

- The MTLB size is multiplied by TBUCFG_MTLB_PARTS
- The mtlbpart field defines the $\log_2$(TBUCFG_MTLB_PARTS) most significant index bits

When direct indexing is enabled for a TBU:

- Lookups and updates to the MTLB use the mtlbidx field
- Updates to the MTLB use the way that mtlbway specifies
- Lookups to the MTLB operate on all ways simultaneously

To maintain system performance, Arm recommends that you disable DVM invalidation on TBUs on which direct indexing is enabled. Disable DVM invalidation by setting the appropriate TCU_NODE_CTRLn.DIS_DVM bit. See *3.7.1 TCU_CTRL register* on page 3-106.

2.3.4 RAS implementation

Reliability, Availability, and Serviceability (RAS) features enable SRAM corruption to be detected and corrected, optionally generating interrupts into the system. All MMU-700 RAM-based caches support RAS error detection and correction. This section describes RAS implementation in the CoreLink MMU-700 System Memory Management Unit.

MMU-700 implements a combination of *Single-Error-Correct-Double-Error-Detect* (SECDED) and *Double-Error-Detect* (DED) error correction mechanisms.

SECDED is used in RAMs where a double error usually means that the SMMU cannot contain this error and must raise a *Critical Error Interrupt*. DED is used in TLB TAGS or DATA, where a single or double error can be recovered by fetching data from System Memory.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Also, the SMMU raises *Fault Handling Interrupts* (FHIs), *Error Recovery Interrupts* (ERIs), and *CRitical Error Interrupts* (CRIs) based on a contained error or uncontained error respectively.

The following table shows the RAMs in MMU-700, and actions that are taken when errors occur.

Table 2-12 RAM RAS error sources

RAM name	Error correction and detection mechanism	RAS error triggered		RAS interrupts triggered
BIU WDB ROBUFF_D	SEC	CE		FHI
	DED	Poison supported:	DE	FHI
		Poison not supported:	UE (UC)	ERI, FHI, and CRI
BIU WDB ROBUFF_C	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
BIU WDB ROBUFF_P	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU OGQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU UOQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU DTIQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU REQ	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU RSP	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU LB	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB_ENTRY LEFT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB_ENTRY RIGHT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB PTR LEFT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU TOU HLB PTR RIGHT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU DCU MTLB PLIM	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU DCU MTLB PCNT	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI
TLBU DCU MTLB REPL	SEC	CE		FHI
	DED	UE (UC)		FHI, ERI, and CRI

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-12 RAM RAS error sources (continued)

RAM name	Error correction and detection mechanism	RAS error triggered	RAS interrupts triggered
TLBU DCU MTLB TAGS	SED	CE	FHI
	DED	CE	FHI
TLBU DCU MTLB DATA	SED	CE	FHI
	DED	CE	FHI
TMU TWB BSU	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU HZU PTR	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU TWB WMB LKP STATUS	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU TWB WMB WLK STATUS	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU TWB WMB SCRATCH	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU HTTU RAM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC PLIM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC PCNT	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC REPL	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU WCB MWC TAGS	SED	CE	FHI
	DED	CE	FHI
TMU WCB MWC DATA	SED	CE	FHI
	DED	CE	FHI
TMU CCB MCC PLIM	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU CCB MCC PCNT	SEC	CE	FHI, ERI, and FHI on DED
	DED	UE (UC)	FHI, ERI, and CRI
TMU CCB MCC REPL	SEC	CE	FHI
	DED	UE (UC)	FHI, ERI, and CRI
TMU CCB MCC TAGS	SED	CE	FHI
	DED	CE	FHI
TMU CCB MCC DATA	SED	CE	FHI
	DED	CE	FHI

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.3.5 Quality of Service

You can program the TCU with a priority level for each TBU. The priority level is applied to every translation from that TBU.

The TCU uses this priority level to:

- Arbitrate between translations that are waiting in the translation request buffer when translation manager slots become available
- Arbitrate between translation manager slots when they access the caches and perform configuration table walks and translation table walks
- Determine the AXI **AxQOS** value for translation table walks and configuration table walks that the TCU issues on the QTW/DVM interface

The arbiters contain starvation avoidance mechanisms to prevent transactions from being stalled indefinitely.

The TBU does not implement any prioritization between transactions. Arm recommends that bus masters with different QoS requirements use separate TBUs for translation.

Related references

3.7.5 TCU_NODE_CTRLn register on page 3-110
3.7.2 TCU_QOS register on page 3-107

2.3.6 *Distributed Virtual Memory* (DVM) messages

The QTW/DVM interface supports DVM messages. The MMU-700 supports DVMv8.4.

The interface supports DVM transactions of message types TLB Invalidate and Synchronization. The interface accepts all other DVM transaction message types, and sends a snoop response, but otherwise ignores such transactions.

Tie the **sup_btm** input signal HIGH when the system supports Broadcast TLB Maintenance.

When Broadcast TLB Maintenance is supported, you can use SMMU_CR2 and SMMU_S_CR2 to control how the SMMU handles TLB Invalidate operations as follows:

SMMU_CR2.PTM = 0	Non-secure TLB Invalidate operations are applied to the TLBs.
SMMU_CR2.PTM = 1	Non-secure TLB Invalidate operations have no effect.
SMMU_S_CR2.PTM = 0	Secure TLB Invalidate operations are applied to the TLBs.
SMMU_S_CR2.PTM = 1	Secure TLB Invalidate operations have no effect.

Note

When **sup_btm** is tied HIGH, the reset value of SMMU_CR2.PTM and SMMU_S_CR2.PTM is 1.

Note

Although TLB Invalidate operations have no effect when PTM = 1, the QTW/DVM interface still returns the appropriate response.

The QTW/DVM interface might receive DVM Sync transactions without receiving a DVM TLB Invalidate transaction, or when the PTM bits have masked a TLB Invalidate. If no DVM TLB Invalidate operations have occurred since the most recent DVM Sync transaction, subsequent DVM Sync transactions result in an immediate DVM Complete transaction. This behavior ensures that the TCU does not affect system DVM performance unless TLB Invalidate operations are performed.

The DTI interface allocates the access permissions and shareability of DVM Complete transactions as follows:

- **ARPROT** = `0b000`, indicating Unprivileged, Secure, Data access
- **ARDOMAIN** = `0b01`, indicating Inner Shareable

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

For a DVM Operation or DVM Sync request on the **AC** channel, the snoop response signal **CRRESP[4:0]** is always set to `0b00000`.

Related references

3.2 SMMU architectural registers on page 3-88

2.3.7 TCU transaction handling

The transaction width, burst length, and transfer size that the TCU supports depend on the transaction type.

The following table shows the TCU support for read transactions.

Table 2-13 TCU support for read transactions

Transaction type	Transaction width	ARID[n:2]	ARID[1:0]
Level 1 Stream table or Level 1 Context Descriptor table lookup	64-bit	Config slot number	`2'b01`
Stream table or Context Descriptor table lookup	512-bit	Config slot number	`2'b01`
Translation table lookup	64-bit	PTW slot number	`2'b10`
Command queue read	128-bit	All 0	`2'b00`
DVM Complete	-	Bit 2 is 1 and all other bits are 0	`2'b00`

DVM Complete transactions are always one beat of full data width.

Command queue reads and DVM Complete transactions are independent of translation slots. Therefore, the maximum number of read transactions that the TCU can issue at any time is `TCUCFG_PTW_SLOTS` + 2.

The following table shows the TCU support for write transactions.

Table 2-14 TCU support for write transactions

Transaction type	Transaction width	AWID[1:0]
Event queue write	256-bit	`2'b00`
PRI queue write	128-bit	`2'b00`
Message Signaled Interrupt (MSI)	32-bit	`2'b00`
HTTU write	128-bit	`2'b11`

Only one write transaction can be outstanding at a time.

All read and write transactions are aligned to the transaction size.

2.3.8 TCU prefetch

The TCU can prefetch translations on a per-context basis to improve translation performance for real-time masters that access memory linearly. If TCU prefetch is enabled, a second translation request occurs after the original request. This second translation request is regarded as the *prefetch* because it is an advance request of the next translation that is expected to be requested. This second request is Speculative and is used to allocate into the caches of the TCU.

Software can enable TCU prefetch for a particular translation context by programming the *Stream Table Entry* (STE). Bits [121:120] are IMPLEMENTATION DEFINED in the SMMUv3 architecture. See the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

The MMU-700 uses these bits for the PF field as follows:

PF, bits [121:120]

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

This field determines whether prefetch is enabled or disabled for the translation context that this STE defines as follows:

`0b00`	Prefetching disabled
`0b01`	Reserved
`0b10`	Forward prefetching
`0b11`	Backward prefetching

Prefetching disabled
TCU prefetch does not occur

Reserved
Reserved values must not be used

Forward prefetching
The address to be prefetched is the first address following the end of the translation range, as DTI_TBU_TRANS_RESP.TRANS_RNG/DTI_ATS_TRANS_RESP.TRANS_RNG indicates

Backward prefetching
The address to be prefetched is the last address before the beginning of the translation range, as DTI_TBU_TRANS_RESP.TRANS_RNG/DTI_ATS_TRANS_RESP.TRANS_RNG indicates

Whenever a miss occurs in the micro TLB and Main TLB of the TBU, the TBU sends a translation request to the TCU. If the STE for the translation is programmed to enable prefetch, each translation request to the TCU can also potentially result in a prefetch that occurs after the original request is complete. When each incoming translation request completes its translation in the TCU, the STE.PF field indicates whether TCU prefetch is enabled. If TCU prefetch is enabled, a second translation request, the prefetch request, is then issued into the same TCU translation slot. This prefetch request is Speculative, and only allocates into the TCU walk caches. A translation response for the prefetch is not returned to the TBU.

When the TCU handles each incoming translation request from the TBU, translation table walks might or might not occur depending on whether there is a hit in each level of walk cache that is looked up. Translation table walks also might or might not occur for the subsequent prefetch request. The number of memory accesses that are performed for this prefetch are unrelated to the number of memory accesses that are performed for the original translation request.

Consider the following examples:

1. An incoming translation request might hit in the lowest level of walk cache, but the subsequent prefetch request might still require at least one translation table walk to memory.
2. The original translation request might require multiple translation table walks, but the subsequent prefetch request might hit in the lowest level of walk cache and not require any memory accesses. If the prefetch request hits in the lowest level of walk cache, then the walk caches are not updated and no memory accesses are performed.

   ——— **Note** ———

   The walk cache uses a round-robin replacement policy..

   ———————————

The prefetch can only occur when the original request is complete irrespective of whether translation table walks were required. The prefetch must wait for completion because it uses the same translation slot as the original request. Waiting for completion of the original request means that by the time it becomes possible for the prefetch to be initiated, the TCU might have already received a non-speculative request for the next translation and begun to handle this request using a separate translation slot. Therefore, TCU prefetch only results in a performance advantage if the number of cycles between each sequential translation request from the TBU is greater than the number of cycles that is taken for the TCU to handle the original translation request and to start the subsequent prefetch.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Even if TCU prefetch is enabled, a prefetch does not occur if one of the following caused the original request:

- A Speculative translation request, that is, **DTI_TBU_TRANS_REQ.PERM[1:0]** = `2'b11`, if a TBU receives a StashOnceShared, StashOnceUnique, or StashTranslation transaction
- A translation request for an atomic transaction that provides a data response, that is, **DTI_TBU_TRANS_REQ.PERM[1:0]** = `2'b10`, if a TBU receives an AtomicLoad, AtomicSwap, or AtomicCompare transaction

If the original translation request returns one the following, prefetch also does not occur:

- Fault response
- Global bypass response
- Stream bypass response

Note

Prefetch applies to both ATS and non-ATS translation requests.

2.3.9 Error responses

AMBA defines external AXI slave error, SLVERR, and external AXI decode error, DECERR. The MMU-700 error response behavior depends on the interface.

The TCU QTW/DVM interface treats SLVERR and DECERR identically, as an abort.

When terminating a transaction, the TBS interface generates a SLVERR response.

If the TBU TBM interface receives a SLVERR or DECERR response to a downstream transaction, it propagates the same abort type to the TBS interface.

2.3.10 Conversion between ACE-Lite and Armv8 attributes

The SMMUv3 architecture defines attributes in terms of the Armv8 architecture. See the *Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile*. The MMU-700 components are therefore required to perform conversion between ACE-Lite and Armv8 attributes.

The TBU must convert:

- ACE-Lite attributes to Armv8 attributes when it receives transactions on the *Transaction Slave* (TBS) interface
- Armv8 attributes to ACE-Lite attributes when it outputs transactions on the *Transaction Master* (TBM) interface

The TCU must convert Armv8 attributes to ACE-Lite attributes when it outputs transactions on the QTW/DVM interface.

Slave interface memory type attribute handling

The memory attributes that apply to the TBS interface are contained in the **AxCACHE** and **AxDOMAIN** signals.

The following table shows the ACE-Lite to Armv8 attribute conversions that the TBU TBS interface performs.

Table 2-15 MMU-700 ACE-Lite to Armv8 memory attribute conversions

AxCACHE attribute	AxDOMAIN attribute	Armv8 memory attribute	Armv8 Shareability
Device Non-bufferable	System	Device-nGnRnE	Outer Shareable
Device Bufferable	System	Device-nGnRE	Outer Shareable

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-15 MMU-700 ACE-Lite to Armv8 memory attribute conversions (continued)

AxCACHE attribute	AxDOMAIN attribute	Armv8 memory attribute	Armv8 Shareability
Normal Non-cacheable Bufferable Normal Non-cacheable Non-bufferable Write-Through No Allocate Write-Through Read-Allocate Write-Through Write-Allocate Write-Through Read and Write-Allocate	Any	Normal Inner Non-cacheable Outer Non-cacheable	Outer Shareable
Write-Back No Allocate Write-Back Read-Allocate Write-Back Write-Allocate Write-Back Read-Allocate Write-Allocate	Non-shareable Inner Shareable Outer Shareable	Normal Inner Write-Back Outer Write-Back	Non-shareable Non-shareable Outer Shareable

Note

- Write-Back transactions are always treated as non-transient
- The Armv8-A Read-Allocate and Write-Allocate hints are the same as the hints that the **AxCACHE** Write-Back type provides
- The TBU TBS interface converts instruction writes into data writes, that is, it treats **awprot_s[2]** as 0

Master interface memory type attribute handling

The memory attributes that apply to the TBM and the QTW/DVM interfaces are contained in the **AxCACHE** and **AxDOMAIN** signals.

In addition, the TBU TBM interface can use the **AxLOCK** signal to indicate an Exclusive access. The QTW/DVM interface does not use the **AxLOCK** signal.

On the TBU TBM interface, a bit on **AxUSER** indicates whether the memory type before the conversion is Outer Cacheable.

The following table shows the Armv8 to ACE-Lite attribute conversions that the master interfaces perform.

Table 2-16 MMU-700 Armv8 to ACE-Lite memory attribute conversions

Armv8 memory attribute	AxCACHE attribute	AxDOMAIN attribute	AxLOCK attribute	AxUSER Outer Cacheable
Device-nGnRnE	Device Non-bufferable	System	As *Transaction Slave* (TBS) **AxLOCK** value	0
Device-GRE Device-nGRE Device-nGnRE	Device Bufferable	System	As TBS **AxLOCK** value	0

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-16 MMU-700 Armv8 to ACE-Lite memory attribute conversions (continued)

Armv8 memory attribute	AxCACHE attribute	AxDOMAIN attribute	AxLOCK attribute	AxUSER Outer Cacheable
Normal Inner Non-cacheable Outer Non-cacheable Normal Inner Write-Through Outer Non-cacheable Normal Inner Write-Back Outer Non-cacheable	Normal Non-cacheable Bufferable	System	As TBS **AxLOCK** value	0
Normal Inner Non-cacheable Outer Write-Through Normal Inner Write-Through Outer Write-Through Normal Inner Write-Back Outer Write-Through Normal Inner Non-cacheable Outer Write-Back Normal Inner Write-Through Outer Write-Back	Normal Non-cacheable Bufferable	System	As TBS **AxLOCK** value	1
Normal Inner Write-Back Outer Write-Back	Write-Back No Allocate Write-Back Read-Allocate Write-Back Write-Allocate Write-Back Read and Write-Allocate	If **AxBURST** == FIXED, Non-shareable. If **AxBURST** != FIXED, the attribute reflects the Armv8 Shareability: • Non-shareable • Inner Shareable • Outer Shareable	0	1

Related references
2.3.11 AXI USER bits that MMU-700 TBU defines on page 2-55

2.3.11 AXI USER bits that MMU-700 TBU defines

The TBU TBM interface **AxUSER** signals, **aruser_m** and **awuser_m**, have 5 bits more than `TBUCFG_AxUSER_WIDTH` defines. These extra bits are output in higher-order bits of **aruser_m** and **awuser_m**.

The following table shows the MMU-700-defined **aruser_m** and **awuser_m** bits, where *w* represents the AXI USER bus width that `TBUCFG_AxUSER_WIDTH` defines.

Table 2-17 MMU-700-defined aruser_m and awuser_m bits

Bit position	Value
[*w*+4]	Outer Cacheable
[*w*+3:*w*]	The IMPLEMENTATION DEFINED page-based hardware attributes

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Page Based Hardware Attribute (PBHA) in SMMUs

The Arm architecture defines that 4 bits in both stage 1 and stage 2 leaf page table entry formats are reserved for software use. Armv8.4 and SMMUv3.2 define a mechanism where software can declare that it does not require them, on a per bit basis.

See the following:

- *Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile*
- *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*

The *Page Based Hardware Attribute* (PBHA) mechanism effectively hands over control of those bits to IMPLEMENTATION DEFINED hardware purposes. These bits are called PBHA bits.

For more information about PBHA bits, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

Related references

Master interface memory type attribute handling on page 2-54

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.4 Constraints and limitations of use

Certain usage constraints and limitations apply to the MMU-700.

Unless otherwise specified, an IMPLEMENTATION DEFINED field in a structure that the MMU-700:

- Generates is 0
- Reads is ignored

This section contains the following subsections:

2.4.1 SMMUv3 implementation

This section describes SMMUv3 implementation in the CoreLink MMU-700 System Memory Management Unit.

ID register architectural options

This section describes ID register architectural options in the CoreLink MMU-700 System Memory Management Unit.

The following table shows the architectural options for MMU-700 from the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* that the SMMUv3 ID registers expose.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-18 SMMUv3 ID registers options

Register	Field	Value	Description
SMMU_IDR0	S2P	1	Stage 2 translation supported
	S1P	1	Stage 1 translation supported
	TTF	11	AArch64 and AArch32 translation supported
	COHACC	**sup_cohacc**	Coherent accesses supported, system configuration option
	BTM	**sup_btm**	Broadcast TLB maintenance, system configuration option
	HTTU[1:0]	{**sup_httu**, 1'b0}	Access and dirty flag update supported
	DORMHINT	0	Dormant hint is not supported
	Hyp	1	EL2-E2H is supported
	ATS	1	ATS is supported
	NS1ATS	0	Stage 1-only ATS is supported
	ASID16	1	16-bit ASID is supported
	MSI	1	*Message Signaled Interrupts* (MSIs) are supported
	SEV	**sup_sev**	Send event is supported, system configuration option
	ATOS	0	ATOS is not supported
	PRI	1	PRI is supported
	VMW	1	VMID wildcard matching supported
	VMID16	1	16-bit VMIDs are supported
	CD2L	1	2-level context descriptor tables are supported
	VATOS	0	Virtual ATOS is not supported
	TTENDIAN	2'b00	Mixed-endian translation walks are supported
	STALL_MODEL	{1'b0, SMMU_S_CR0.NSSTALLD}	Stall and terminate models that are supported unless the Secure world disables Non-secure stalling
	TERM_MODEL	0	Terminating a transaction with RAZ/WI is supported
	ST_LEVEL	01	2-level stream table is supported
SMMU_IDR1	SIDSIZE	24	24-bit stream IDs are supported
	SSIDSIZE	20	20-bit substream IDs are supported
	PRIQS	5'b10011	2^{19} PRI queue entries are supported
	EVENTQS	5'b10011	2^{19} Event queue entries are supported
	CMDQS	5'b10011	2^{19} Command queue entries are supported
	ATTR_PERMS_OVR	1	Incoming permission attributes can be overridden
	ATTR_TYPES_OVR	1	Incoming memory attributes can be overridden
	REL	0	N/A, not fixed base addresses
	QUEUES_PRESET	0	Not fixed queue base addresses
	TABLES_PRESET	0	Not fixed table base addresses
SMMU_IDR2	BA_VATOS	0	N/A, no VATOS support

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-18 SMMUv3 ID registers options (continued)

Register	Field	Value	Description
SMMU_IDR3	HAD	1	Hierarchical attribute disable supported
	PBHA	1	Page-based hardware attributes are supported
	XNX	1	EL0/EL1 stage 2 execute control is supported
	PPS	1	PASID, when present always used in PRI auto-generated response
	MPAM	1	MPAM is supported
	FWB	1	S2 control of memory type is supported
	STT	1	Small translation tables are supported
	RIL	1	Range-based invalidation and Level hint are supported
	BBML	2	Break before Make level 2 is supported
SMMU_IDR4	IMPDEF	0	No IMPLEMENTATION DEFINED features
SMMU_IDR5	OAS	**sup_oas**	Output address size, system configuration option
	GRAN4K	1	4K translation granule is supported
	GRAN16K	1	16K translation granule is supported
	GRAN64K	1	64K translation granule is supported
	VAX	01	Virtual addresses of 52 bits per CD.TTBx are supported
	STALL_MAX	TCUCFG_XLATE_SLOTS	Maximum number of outstanding stalled transactions
SMMU_IIDR	Implementor	0x43B	Arm implementation
	Revision	MAX(*p_level*, ecorevnum)	Where *p_level* is: 0 For p0 1 For p1
	Variant	0	r0
	ProductID	0x487	Second SMMUv3 implementation from Arm
SMMU_AIDR	ArchMinorRev	2	Arch rev 2
	ArchMajorRev	0	SMMUv3
SMMU_S_IDR0	MSI	1	Secure MSIs are supported
	STALL_MODEL	2'b00	Stall and terminate model is supported
SMMU_S_IDR1	S_SIDSIZE	24	24-bit Secure stream IDs are supported
	SEL2	1	Secure EL2 is supported
	SECURE_IMPL	1	Two Security states are implemented
SMMU_S_IDR3	SAMS	1	Secure ATS maintenance is not implemented
SMMU_S_IDR4	IMPDEF	0	No IMPLEMENTATION DEFINED features

Non-implemented commands and events

This section describes the non-implemented commands and events in the CoreLink MMU-700 System Memory Management Unit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Event queue

MMU-700 does not generate the following events:

- F_UUT
- F_TLB_CONFLICT
- F_CFG_CONFLICT
- E_PAGE_REQUEST
- IMPDEF_EVENTn

Command queue

The following commands are accepted but silently ignored:

- CMD_PREFETCH_CONFIG
- CMD_PREFETCH_ADDR
- CMD_CFGI_VMS_PIDM

The CMD_ATC_INV command is supported for the Non-secure Command queue only. If the TCU encounters this command in the Secure Command queue, it results in a Secure Command queue error with reason code CERROR_ILL.

IMPLEMENTATION DEFINED fields

This section describes the IMPLEMENTATION DEFINED fields in the CoreLink MMU-700 System Memory Management Unit.

Unless otherwise specified, IMPLEMENTATION DEFINED fields in structures that MMU-700:

- Generates are 0
- Reads are ignored

Non-implemented registers

This section describes the non-implemented registers in the CoreLink MMU-700 System Memory Management Unit.

The following optional registers are not implemented and are RAZ/WI:

- SMMU_IDR4
- SMMU_STATUSR
- SMMU_GATOS_*
- SMMU_S_GATOS_*
- SMMU_VATOS_*

The following PMCG registers are not implemented and are RAZ/WI:

- SMMU_PMCG_IRQ_CFG0
- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2

Related references

3.2 SMMU architectural registers on page 3-88

2.4.2 AMBA implementation

This section describes AMBA implementation in the CoreLink MMU-700 System Memory Management Unit.

ACE-Lite feature support

The CoreLink MMU-700 System Memory Management Unit supports many ACE-Lite features.

The following table shows the ACE-Lite features that the CoreLink MMU-700 System Memory Management Unit supports.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-19 ACE-Lite feature support

Specification issue	Interface properties	ACE-Lite TBU	TCU
E	Ordered_Write_Observation	Y	N
F.2	Wakeup_Signals	Y	Y
	Check_Type	N	N
	Poison	Y	Y
	Trace_Signals	N	N
	QoS_Accept	N	N
	Loopback_Signals	Y	N
	Untranslated_Transactions	Y	N
	NSAccess_Identifiers	N	N
	Persist_CMO	Y	N
	DVM_v8.1	-	Y
	Coherency_Connection_Signals	-	Y
	Cache_Stash_Transactions	Y	N
	Atomic_Transactions	Y	Y
	DeAllocation_Transactions	Y	N
	WriteEvict_Transaction	N	N
	Barrier_Transactions	N	N
G	MPAM_Support	Y	Y
	Unique_ID_Support	Y	Y
	Check_Type (Odd_Parity_Byte_All)	N	N
	Read_Data_Reordering	Y	Y
	Partial_Read_Data	Y	N
	Persistent CMO Enhancements	N	N

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-19 ACE-Lite feature support (continued)

Specification issue	Interface properties	ACE-Lite TBU	TCU
H	DVMv8.4	N	Y
	Memory Tagging	Y	N
	Read_Interleaving_Disabled	Y	Y
	Transaction Response Extensions	Y	Y
	Untranslated_Transactions Extensions	Y	N
	CMO_On_Read	N	N
	CMO_On_Write	N	N
	Read_Data_Chunking	Y	N
	Write_Plus_CMO	N	N
	MTE_Support	N	N
	Prefetch_Transaction	Y	Y
	WriteZero_Transaction	N	N
	Consistent_DECERR	N	N
	Exclusive_Accesses	Y	N
	Max_Transaction_Bytes	N	N
	Regular_Transactions_Only	N	N
	Shareable_Transactions	Y	Y

SLVERR and DECERR

This section describes SLVERR and DECERR in the CoreLink MMU-700 System Memory Management Unit.

The TCU QTW interface treats SLVERR and DECERR identically, as an abort.

The TBU TBS interface generates SLVERR when terminating a transaction that requires an abort response.

If the TBU TBM interface receives an SLVERR or DECERR response to a downstream transaction, the same abort type is propagated to the TBS interface.

Attribute handling

This section describes attribute handling in the CoreLink MMU-700 System Memory Management Unit.

When translation is enabled and a PCIe Root Complex issues transactions to a TBU, the following apply, depending on the type of transaction:

Untranslated (non-ATS) transaction

The SMMU applies attributes that a combination of the input attributes, STE overrides, and translation table descriptors determine.

Fully-translated (full ATS) transaction

The SMMU does not modify the attributes that are encoded in the fully translated transaction. The unmodified attributes are used as the output attributes.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Partially-translated (Split-stage ATS) transaction

The ATS translation response from the TCU to the PCIe Root Complex includes Stage 1 and Stage 2 attributes. The Stage-1-translated transaction to the TBU encodes these Stage 1 and Stage 2 attributes. The SMMU performs Stage 2 translation and combines the Stage 2 attributes a second time, but this does not affect the output attributes. The output attributes remain the same as the attributes that the TBU receives for the Stage-1-translated transaction.

For information about the preceding transactions and their attributes, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

Note

TBUs that are connected to a PCIe Root Complex must have the **pcie_mode** input signal tied HIGH, as the table in *A.2.9 TBU tie-off signals* on page Appx-A-192 describes.

Slave interface attribute handling

This section describes the slave interface attribute handling in the CoreLink MMU-700 System Memory Management Unit.

The TBU TBS interface converts the incoming ACE-Lite attributes into Armv8 attributes.

The following table shows the slave interface attribute handling.

Table 2-20 Slave interface attribute handling

ACE-Lite AxCACHE	ACE-Lite AxDOMAIN	Armv8 memory type	Armv8 shareability	Description
Device Non-bufferable	SY	Device-nGnRnE	OSH	-
Device Bufferable	SY	Device-nGnRE	OSH	-
Normal Non-cacheable Bufferable, Normal Non-cacheable Non-bufferable, Write-Through No-allocate, Write-Through Read-Allocate, Write-Through Write-Allocate, Write-Through Read and Write-Allocate	Any	Normal-iNC-oNC	OSH	Normal Non-cacheable Non-bufferable is a deprecated AxCACHE type and is converted to Normal Non-cacheable Bufferable. Write-Through types are converted to Non-cacheable on input to match the normalization step on output.
Write-back No-allocate, Write-back Read-Allocate, Write-back Write-Allocate, Write-back Read and Write-Allocate	NSH/ISH/OSH	Normal-iWB-oWB	NSH/OSH	The Armv8 RA and WA hints depend on the Write-Back type. The transaction is always treated as non-transient. All ISH Shareability types are converted to OSH.

Slave interface AxPROT handling

Instruction writes are converted into data writes on the TBU TBS interface. In effect, **AWPROT[2]** is ignored and always treated as 0.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Master interface attribute handling

This section describes the master interface attribute handling in the CoreLink MMU-700 System Memory Management Unit.

Normalization

Both AMBA master interfaces, TBU TBM interface and TCU QTW interface, carry normalized attributes using the standard Cortex Armv8 scheme:

- Memory that is marked as Inner Write-Back Cacheable and Outer Write-Back Cacheable is output as Write-Back Cacheable
- Memory that is marked as Inner Non-cacheable or Write-Through Cacheable, or Outer Non-cacheable or Write-Through Cacheable, is output as Non-cacheable, Outer Shareable

On the TBU TBM interface, a bit on **AxUSER** indicates whether the output memory type before this conversion is outer cacheable.

The following table shows how the Armv8 transaction types are translated into AMBA ACE-Lite signals.

Table 2-21 Armv8 transaction types translated into AMBA ACE-Lite signals

Armv8 Memory Type	AxCACHE (TBM, QTW)	AxDOMAIN (TBM, QTW)	AxLOCK (TBM)	AxUSER outer cacheable bit (TBM)
Device-nGnRnE	Device No-bufferable	SY	TBS **AxLOCK** value	`0`
Device-GRE, Device-nGRE, Device-nGnRE	Device Bufferable	SY	TBS **AxLOCK** value	`0`
Normal-iNC-oNC, Normal-iWT-oNC, Normal-iWB-oNC	Normal Non-cacheable Bufferable	SY	TBS **AxLOCK** value	`0`
Normal-iNC-oWT, Normal-iWT-oWT, Normal-iWB-oWT, Normal-iNC-oWB, Normal-iWT-oWB	Normal Non-cacheable Bufferable	SY	TBS **AxLOCK** value	`1`
Normal-iWB-oWB	Write-back No-allocate / Write-back Read-Allocate / Write-back Write-Allocate / Write-back Read and Write-Allocate, depending on the Armv8 outer allocate hints.	**AxBURST** = FIXED: NSH **AxBURST** != FIXED: NSH or OSH, depending on the Armv8 Shareability	`0`	`1`

AxCACHE encodings

Where there are multiple legal values for **AxCACHE** as the *AMBA® AXI and ACE Protocol Specification* describes, the canonical, non-bracketed, one is used. Therefore, only the **AxCACHE** encodings that the following table shows are used.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-22 AxCACHE encodings

AMBA memory type	ARCACHE	AWCACHE
Device Non-bufferable	0000	0000
Device Bufferable	0001	0001
Normal Non-cacheable Bufferable	0011	0011
Write-back No-allocate	1011	0111
Write-back Read-Allocate	1111	0111
Write-back Write-Allocate	1011	1111
Write-back Read and Write-Allocate	1111	1111

AxREGION

This section describes the AxREGION in the CoreLink MMU-700 System Memory Management Unit.

This signal is not required.

- On QTW, it is driven as 0
- On TBM, it reflects the value of the corresponding TBS transaction, or is driven as 0 if the transaction originated in the TBU

DVM interface

This section describes the DVM interface in the CoreLink MMU-700 System Memory Management Unit.

Supported DVM operations

In response to an **ACSNOOP** request, the CRRESP field is always driven as `0b00000`.

All DVM operations are handled in a protocol-compliant manner, because the interconnect does not know that the TCU does not need DVM operations other than TLB invalidate. Any DVM operation with a DVM Message Type in **ACADDR[14:12]** other than TLB Invalidate or Synchronization is accepted and responded to on the CR channel but otherwise ignored.

DVM complete

DVM Complete messages are presented with:

- **ARPROT[2:0]** = `0b000`, that is, Unprivileged Secure Data
- **ARDOMAIN[1:0]** = `0b10`, that is, Outer Shareable

Internally terminated transactions

This section describes the internally terminated transactions in the CoreLink MMU-700 System Memory Management Unit.

Transactions that are terminated inside the TBU are returned with all **RUSER** and **BUSER** bits zero.

Transaction types

This section describes the transaction types in the CoreLink MMU-700 System Memory Management Unit.

MMU-700 supports several special transaction types, distinguished by a nonzero encoding of **AxSNOOP**. This section describes how each transaction type is handled.

Unless otherwise specified, transactions are propagated on TBM with the same transaction type that was presented on TBS.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

An *ordinary read* or *ordinary write* is one with **AxSNOOP** = `0b0000`, that is, depending on **AxDOMAIN**, and whether it is a read or a write, one of the following:

- ReadNoSnoop
- ReadOnce
- WriteNoSnoop
- WriteUnique

In the *AMBA® LTI Protocol Specification*, see:

- *Section 5.2* for information about the mapping of LTI transactions to AXI types
- *Section 6* for information about handling LTI responses to convert them to AXI responses

Transactions that can result in a translation fault

In an MMU-700 system, some transactions can result in a translation fault, and certain behavior is associated with such transactions.

The MMU-700 treats the following transactions as ordinary reads when calculating translation faults:

- CleanShared
- CleanInvalid
- MakeInvalid
- CleanSharedPersist
- ReadOnceMakeInvalid
- ReadOnceCleanInvalid

Therefore, these transactions might require either read permission or execute permission at the appropriate privilege level.

The MMU-700 treats the following transactions as ordinary writes when calculating translation faults:

- WriteUniquePtlStash
- WriteUniqueFullStash

Therefore, these transactions require write permission at the appropriate privilege level.

CleanShared, CleanInvalid, MakeInvalid, and CleanSharedPersist transactions do not have a memory type. The input transaction and output transaction memory type and allocation hints are ignored and replaced by Normal, Inner Write-Back, Outer Write-Back, Read Allocate, Write Allocate. This behavior means that the **ARDOMAIN** output on the TBM interface is never System Shareable for these transactions, because they are never Non-cacheable or Device.

The MMU-700 treats transactions that pass the translation fault check as follows:

MakeInvalid transactions

The MMU-700 converts MakeInvalid transactions to CleanInvalid transactions, unless the translation also grants write permission and *Destructive Read Enable* (DRE) permission.

ReadOnceMakeInvalid and ReadOnceCleanInvalid transactions

The MMU-700 outputs ReadOnceMakeInvalid transactions as ReadOnceCleanInvalid transactions, unless the translation also granted write permission and DRE permission.
If the final transaction attributes on the TBU TBM interface are not Inner Shareable Write-Back or Outer Shareable Write-Back, the MMU-700 converts ReadOnceMakeInvalid and ReadOnceCleanInvalid transactions into ordinary reads.

WriteUniquePtlStash and WriteUniqueFullStash transactions

If they pass the translation fault check, the MMU-700 converts WriteUniquePtlStash and WriteUniqueFullStash transactions to ordinary write transactions if either:

- The translation did not grant *Directed Cache Prefetch* (DCP) permission
- The final transaction attributes on the TBU TBM interface are not Inner Shareable or Outer Shareable Write-Back

If such a conversion occurs, **AWSTASH*** is driven as 0.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Transactions that cannot result in a translation fault

In an MMU-700 system, certain transactions cannot result in a translation fault, and certain behavior is associated with such transactions.

The following transactions never result in a translation fault:

- StashOnceShared
- StashOnceUnique
- StashTranslation

If any of these transactions require a translation request to the TCU, the MMU-700 issues a Speculative translation request on the DTI interconnect. StashOnceShared and StashOnceUnique transactions are terminated in the TBU, with a **BRESP** value of OKAY, when any of the following cases apply:

- The translation did not grant *Directed Cache Prefetch* (DCP) permission
- The final transaction attributes on the TBM interface are not Inner Shareable or Outer Shareable Write-Back
- The translation did not grant any of read, write, or execute permission at the appropriate privilege level

  ——— **Note** ———

  Only one of these permissions is required for the stash transaction to be permitted.

  ———

——— **Note** ———

A **BRESP** value of OKAY indicates transaction success. The MMU-700 always generates this value when a StashOnceShared or a StashOnceUnique transaction is terminated in the TBU. This behavior applies even when a StreamDisable or GlobalDisable translation response causes the transaction to be terminated.

———

The MMU-700 never propagates StashTranslation transactions downstream, and uses StashTranslation only to prefetch Main TLB contents. MMU-700 always terminates StashTranslation transactions with a **BRESP** value of OKAY, even if no translation could be stored in the Main TLB.

The TBU ignores **AWPROT[0]** and **AWPROT[2]** for StashTranslation transactions, because they do not affect Speculative translation requests.

——— **Note** ———

A StashTranslation transaction can be used to prefetch translations into the Main TLB of the MMU-700. However, for this prefetching to be useful, any subsequent transactions that intend to take advantage of the translations that have been prefetched into the Main TLB must use the same StreamID as the original prefetch. The StreamID identifies a translation context. Using a different StreamID for a subsequent transaction means that this subsequent transaction uses a different translation context to the translation that has been prefetched into the Main TLB and might lead to a TLB miss.

———

2.4.3 MPAM implementation

This section describes *Memory System Resource Partitioning and Monitoring* (MPAM) implementation in the CoreLink MMU-700 System Memory Management Unit.

This section describes the MPAM architectural options that MMU-700 uses from the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A*.

Certain MPAM registers are implemented.

Registers that are not described are not implemented.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

MPAM capacity partitioning manages the following:

- TBU MTLB
- TCU configuration cache
- Walk caches

No other mechanism from the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A* is implemented.

TCU MPAM

This section describes TCU *Memory System Resource Partitioning and Monitoring* (MPAM) implementation in the CoreLink MMU-700 System Memory Management Unit.

Internally, RIS is truncated to 1 bit, but externally it is the normal 4 bits.

RIS 0 = WCB

RIS 1 = CCB

Because RIS is internally truncated to 1 bit, there is no illegal RIS. It is therefore not necessary to report fewer controls in the ID registers for illegal RIS. It is also not necessary to RAZ/WI accesses to the control registers for illegal RIS.

The following table shows the TCU MPAM registers that are implemented.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-23 TCU MPAM registers implemented

Register	Field	Value used[c]	Value not used[d]	Description
MPAMF_IDR_LO (0x0000, Shared)	PARTID_MAX	1, 63, 511	1, 63, 511	In the TBU, set to: ($2^{TBUCFG\_PARTID\_WIDTH}$ - 1) In the TCU, set to ($2^{TCUCFG\_PARTID\_WIDTH}$ - 1)
	PMG_MAX	1	1	Two Non-secure Performance Monitoring groups supported per PARTID
	HAS_CCAP_PART	1	0	Supports cache maximum capacity partitioning
	HAS_CPOR_PART	0	0	Cache portion partitioning not supported
	HAS_MBW_PART	0	0	Memory Bandwidth partitioning not supported
	HAS_PRI_PART	0	0	Priority partitioning not supported
	EXT	1	1	EXTended MPAMF_IDR
	HAS_IMPL_IDR	0	0	Does not have IMPLEMENTATION-SPECIFIC partitioning features
	HAS_MSMON	1	0	Supports performance monitoring by matching a combination of PARTID and PMG
	HAS_PARTID_NRW	0	0	Does not have MPAMF_PARTID_NRW_IDR, MPAMCFG_INTPARTID or intPARTID mapping support

[c] Indicates the values when Resource is in use.
[d] Indicates the values when Resource is not in use.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-23 TCU MPAM registers implemented (continued)

Register	Field	Value used[c]	Value not used[d]	Description
MPAMF_IDR_HI (0x0004, Shared)	HAS_RIS	1	0	Has Resource Instance Selector
	NO_IMPL_PART	1	0	There are no IMPLEMENTATION DEFINED resource controls that MPAMF_IMPL_IDR defines
	NO_IMPL_MSMON	1	0	There are no IMPLEMENTATION DEFINED resource monitors that MPAMF_IMPL_IDR defines
	HAS_EXTD_ESR	0	1	MPAMF_ESR is 64-bits. Not relevant because HAS_ESR is 0.
	HAS_ESR	0	1	MPAMF_ESR and MPAMF_ECR are not implemented
	RIS_MAX	0 or 1	0	Maximum RIS value used in the MSC: In the TBU, set to 0 In the TCU, set to 1
MPAMF_SIDR (0x0008, S-Only)	S_PARTID_MAX	1, 63, 511	1, 63, 511	In the TBU, set to ($2^{TBUCFG\_PARTID\_WIDTH}$ - 1) In the TCU, set to ($2^{TCUCFG\_PARTID\_WIDTH}$ - 1)
	S_PMG_MAX	1	-	Two Secure Performance Monitoring groups supported per PARTID
MPAMF_MSMON_IDR (0x0080, Shared)	MSMON_CSU	1	-	Performance monitor supported for Cache Storage Usage by PARTID and PMG
	MSMON_MBWU	0	-	No performance monitor for Memory Bandwidth Usage by PARTID and PMG
	HAS_LOCAL_CAPT_EVNT	1	-	Has the local capture event generator and the MSMON_CAPT_EVNT register
MPAMF_CCAP_IDR (0x0038, Shared)	CMAX_WD	8	-	256 fractions are supported

[c] Indicates the values when Resource is in use.
[d] Indicates the values when Resource is not in use.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-23 TCU MPAM registers implemented (continued)

Register	Field	Value used[c]	Value not used[d]	Description
MPAMF_CSUMON_IDR (0x0088, Shared)	NUM_MON	4	-	Four monitoring counters are implemented
	HAS_CAPTURE	1	-	Has an MSMON_CSU_CAPTURE register for every MSMON_CSU and supports the capture event behavior
MPAMF_IIDR (0x0018, Shared)	All fields	All fields valid	-	Implementation ID Register
MPAMF_AIDR (0x0020, Shared)	ArchMajorRev ArchMinorRev	0x1 0x10	-	MPAM architecture v1.10
MPAMCFG_PART_SEL (0x0100, Banked)	PARTID_SEL	Bits [TCUCFG_PARTID_WIDTH - 1:0] or [TBUCFG_PARTID_WIDTH - 1:0] valid	-	Can select up to 512 partitions to configure, based on TBUCFG_PARTID_WIDTH, TCUCFG_PARTID_WIDTH. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	RIS	Bits [27:24] valid	-	Resource Instance Selector. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MPAMCFG_CMAX (0x0108, Banked)	CMAX	Bits [15:8] valid	-	Can choose up to 256 fractions. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CFG_MON_SEL (0x0800, Banked)	MON_SEL	Bits [1:0] valid	-	Selects the monitor to configure. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	RIS	Bits [27:24] valid	-	Resource Instance Selector. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.

[c] Indicates the values when Resource is in use.
[d] Indicates the values when Resource is not in use.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-23 TCU MPAM registers implemented (continued)

Register	Field	Value used[c]	Value not used[d]	Description
MSMON_CFG_CSU_FLT (`0x0810`, Banked)	PARTID	Bits [`TCUCFG_PARTID_WIDTH - 1:0`] or [`TBUCFG_PARTID_WIDTH - 1:0`] valid	-	Can select up to 512 partitions to configure, based on `TCUCFG_PARTID_WIDTH`, `TBUCFG_PARTID_WIDTH`. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	PMG	`0`	-	Can select up to PMG number 1. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CFG_CSU_CTL (`0x0818`, Banked)	EN	Valid field.	-	The monitor instance is enabled or disabled to collect information. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	CAPT_EVNT	`3'b111`	-	Capture occurs when a MSMON_CAPT_EVNT register is written. All other values are not supported. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	CAPT_RESET	RES0.	-	There is no reason to ever reset a CSU monitor. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	OFLOW_STATUS	RES0	-	Overflow is not possible for a CSU monitor. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.

[c] Indicates the values when Resource is in use.
[d] Indicates the values when Resource is not in use.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-23 TCU MPAM registers implemented (continued)

Register	Field	Value used[c]	Value not used[d]	Description
MSMON_CFG_CSU_CTL (0x0818, Banked)	OFLOW_INTR	RES0	-	This MPAM implementation does not support OFLOW_INTR. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	OFLOW_FRZ	RES0	-	Overflow is not possible for a CSU monitor. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
	SUBTYPE	RES0	-	This field is reserved for future use. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CSU (0x0840, Banked)	All fields	All fields valid	-	Cache storage usage value. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MON_CSU_CAPTURE (0x0848, Banked)	All fields	All fields valid	-	Capture cache storage usage. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.
MSMON_CAPT_EVNT (0x0808, Banked)	All fields	All fields valid	-	Capture event. When direct index or direct partitioning is enabled, this register does not reflect any meaningful value.

TBU MPAM

This section describes TBU *Memory System Resource Partitioning and Monitoring* (MPAM) implementation in the CoreLink MMU-700 System Memory Management Unit.

When TBUCFG_MTLB_DEPTH == 0, the resource is not present, and when TBUCFG_DIRECT_IDX == 1, the resource is present but does not have *Memory System Resource Partitioning and Monitoring* (MPAM) controls. The associated ID Registers must report values of limited control under these circumstances. Therefore, many non-ID control registers are RAZ/WI in such circumstances.

[c] Indicates the values when Resource is in use.
[d] Indicates the values when Resource is not in use.
[c] Indicates the values when Resource is in use.
[d] Indicates the values when Resource is not in use.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

For information about the TBU MPAM registers that are implemented, see the table in *TCU MPAM* on page 2-68.

2.4.4 LTI implementation

This section describes *Local Translation Interface* (LTI) implementation in the CoreLink MMU-700 System Memory Management Unit.

The parameters in the following table are chosen for LTI.

Table 2-24 LTI parameters

Name	Value	Description
LTI_VC_COUNT	2	Two LTI Virtual channels are chosen, one for read and one for write
LTI_ID_WIDTH	TBUCFG_ID_WIDTH	Equal to ID width of incoming transaction
LTI_SID_WIDTH	TBUCFG_SID_WIDTH	Equal to width of incoming SID
LTI_OG_WIDTH	TBUCFG_LTI_OG_WIDTH	Equal to width of incoming Ordering groups
LTI_TLBLOC_WIDTH	-	Width of TLB location, in bits
LTI_LOOP_WIDTH	-	For LTI TBU, the value is equal to TBUCFG_LTI_LOOP_WIDTH. For ACE-Lite TBU, the value is a sum of a constant that must be defined, plus all the User signal widths on the AXI interface.
LTI_LAUSER_WIDTH	0	The LRUSER single bit indicates whether that transaction has been filtered for performance monitoring purposes
LTI_LRUSER_WIDTH	1	
LTI_LCUSER_WIDTH	0	

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.5 Configuration options and methodology

The TBU, TCU, and BAS components in the CoreLink MMU-700 System Memory Management Unit are delivered as SystemVerilog that you can parameterize. No rendering step is required to configure these components. There are several versions of the switch component, part of the BAS components that are delivered, to accommodate different numbers of slave interfaces.

This section contains the following subsections:

2.5.1 TCU I/O configuration options

You can configure the TCU I/O.

The following table shows the TCU I/O configuration options.

Table 2-25 TCU I/O configuration options

Interface and module name	Configuration name	Options	Description
QTW	TCUCFG_QTW_DATA_WIDTH	64, 128, 256, 512	ACE-Lite_DVM interface data width. **Note** The same width is used for walk cache entries.

2.5.2 TCU buffer configuration options

You can configure the TCU buffer.

The following table shows the TCU buffer configuration options.

Table 2-26 TCU buffer configuration options

Configuration name	Options	Description
TCUCFG_CC_DEPTH	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Configuration cache depth, in entries
TCUCFG_WC_DEPTH	8, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536	Walk cache depth, in entries **Note** (TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS)/ TCUCFG_WC_WAYS) must be > 1
TCUCFG_WC_BANKS	1, 2, 4	Number of banks in Walk Cache **Note** (TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS)/ TCUCFG_WC_WAYS) must be > 1

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-26 TCU buffer configuration options (continued)

Configuration name	Options	Description
TCUCFG_WC_WAYS	4, 8, 16	Number of ways in walk cache **Note** (TCUCFG_WC_DEPTH/TCUCFG_WC_BANKS)/ TCUCFG_WC_WAYS) must be > 1
TCUCFG_NUM_TBU	14, 62	Maximum number of DTI masters, that is, DTI-TBU and DTI-ATS masters, that the TCU supports. The value is two less than 16/64 to better fit into system memory maps. **Note** The ACE-Lite TBU and LTI TBU are both examples of DTI-TBU masters. Integration TBU components count as two DTI masters because they contain two ACE-Lite TBUs.
TCUCFG_XLATE_SLOTS	4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096	Total permitted translation requests from all DTI masters **Note** This value must be greater than or equal to TCUCFG_PTW_SLOTS.
TCUCFG_PTW_SLOTS	2, 4, 8, 16, 32, 64, 128, 256, 512	Number of parallel translation table walks
TCUCFG_CTW_SLOTS	1, 2, 4. **Note** This value must not be greater than TCUCFG_PTW_SLOTS.	Number of parallel configuration table walks
TCUCFG_WC_LKP_SLOTS	2-28	Walk Cache Lookup slots. The number of lookup slots that the walk cache uses. If you do not specify a value, the default value is used to provide the best performance when one page size is active in the walk cache. Reduce the value if TCU performance is not critical and increase the value if more than one page size is active in the walk cache. Make sure that the value is not greater than TCUCFG_PTW_SLOTS. You can: • Increase the value of this parameter to improve performance, but with greater area • Decrease the value of this parameter to save area but with reduced performance
TCUCFG_CC_IDXGEN_MODE	0, 1	Index generation mode for the configuration cache 0 Polynomial. Polynomial is the recommended setting for most systems. 1 Simple.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-26 TCU buffer configuration options (continued)

Configuration name	Options	Description
TCUCFG_DTI_ATS	0, 1, 2, 3, 4, 5, 6, 7, 8	Number of DTI-ATS masters **Note** TCUCFG_NUM_TBU is the total number of DTI-TBU and DTI-ATS masters. TCUCFG_DTI_ATS is the total number of DTI-ATS masters.
TCUCFG_PMU_COUNTERS	4, 16, 32	Number of PMU counters
TCUCFG_PARTID_WIDTH	1, 6, 9	Width of PARTID that is supported 1 When set to 1, **PARTID** sent on DTI Interface is all zeros 6 When set to 6, **PARTID[8:6]** sent on DTI Interface is set to 0
TCUCFG_HZU_DEPTH	2, 4, 8, 16, 32, 64.	Number of hazard cache entries
TCUCFG_PREFETCH_SUPPORTED	0, 1	Specifies whether prefetch is supported
TCUCFG_DATARAM_TYPE	0, 1, 2	RAM type for data group of RAMs 0 Two ports. One port is for reads and one port is for writes. 1 One port, that is, one port for both reads and writes. 2 2 × one port, that is, banked configuration. See the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*. **Note** If this parameter is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. Arm recommends that the RAM is implemented as registers in these cases.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-26 TCU buffer configuration options (continued)

Configuration name	Options	Description
TCUCFG_SLOTRAM_TYPE	0, 1, 2	RAM type for slot group of RAMs 0 Two ports. One port is for reads and one port is for writes. 1 One port, that is, one port for both reads and writes. 2 2 × one port, that is, banked configuration. See the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*. **Note** If this parameter is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. Arm recommends that the RAM is implemented as registers in these cases.
TCUCFG_CACHERAM_TYPE	0, 1	RAM type for cache group of RAMs 0 Two ports. One port is for reads and one port is for writes 1 One port, that is, one port for both reads and writes

2.5.3 TCU debug configuration options

You can configure the TCU debug options.

The following table shows the TCU debug configuration options.

Table 2-27 TCU debug configuration options

Configuration name	Options	Description
TCUCFG_USE_ELA_DEBUG	0, 1	Set the TCUCFG_USE_ELA_DEBUG parameter as follows: 0 The **SIGCLKEN<n>**, **SIGNALGRP<n>**, and **SIGQUAL<n>** signals are driven to 0. 1 The **SIGCLKEN<n>**, **SIGNALGRP<n>**, and **SIGQUAL<n>** signals are driven to according to *B.1 TCU observation interfaces* on page Appx-B-204.

2.5.4 ACE-Lite TBU I/O configuration options

You can configure the ACE-Lite TBU I/O.

The following table shows the ACE-Lite TBU I/O configuration options.

Table 2-28 ACE-Lite TBU I/O configuration options

Interface and module name	Configuration name	Options	Description
TBS, TBM	TBUCFG_ID_WIDTH	1-32	AXI ID width.
TBS, TBM	TBUCFG_DATA_WIDTH	64, 128, 256, 512	AXI data width.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-28 ACE-Lite TBU I/O configuration options (continued)

Interface and module name	Configuration name	Options	Description
TBS, TBM	TBUCFG_ARUSER_WIDTH TBUCFG_AWUSER_WIDTH TBUCFG_RUSER_WIDTH TBUCFG_WUSER_WIDTH TBUCFG_BUSER_WIDTH	1-128	AXI USER bus widths.
TBS, TBM	TBUCFG_STASH_SUPPORT	0, 1	Include stash ID signals.
TBS, TBM	TBUCFG_LOOP_WIDTH	1-8	AXI loopback signal width.
TBS, TBM	TBUCFG_WBUF_DEPTH	0, 8, 16, 32, 64, 128, 256, 512, 1024, 2048	Write buffer depth. This parameter selects the maximum number of beats that can be stored in the write buffer. A value of 0 causes the write buffer not to be implemented. For example, for masters where most transactions are reads.
TBS, TBM	TBUCFG_LFIFO_DEPTH	0, 4	Latency FIFO depth. Supported values are as follows: 0, 4.
TBS, TBM	TBUCFG_OT_TRACKER_TYPE	0, 1	Type of the outstanding transaction tracker used. 0 Table. 1 Loopback. Loopback signals to track outstanding transactions. This setting increases the loopback signal width by 2 on the TBM. When using this mode, 4095 outstanding transactions are supported.
TBS, TBM	TBUCFG_ROT_DEPTH	4, 8, 16, 32, 64, 128, 256, 512	Number of outstanding read transactions. This configuration is valid only when TBUCFG_OT_TRACKER_TYPE is 0.
TBS, TBM	TBUCFG_WOT_DEPTH	4, 8, 16, 32, 64, 128, 256, 512	Number of outstanding write transactions. This configuration is valid only when TBUCFG_OT_TRACKER_TYPE is 0.
TBS, TBM	TBUCFG_DATARAM_TYPE	0, 1, 2	RAM type for data group of RAMs. 0 Two ports, that is, one port for reads and one port for writes. 1 One port, that is, one port for both reads and writes. 2 2 × one port, that is, banked configuration. See the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*. **Note** If this parameter is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. Arm recommends that the RAM is implemented as registers in these cases.

2.5.5 TBU register slice configuration options

You can configure the TBU register slice.

The following table shows the TBU register slice configuration options.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-29 TBU register slice configuration options

Configuration name	Configuration options
TBUCFG_SI_AR_HNDSHK_MODE	Supported values are as follows: 0 FULL: Fully registered, double-buffered register slice. 1 FWD: Registered on the forward path only, that is, the direction that **xVALID** on the corresponding interface indicates. 2 REV: Registered on the reverse path only, that is, the direction that **xREADY** on the corresponding interface indicates. 3 BP: Bypass register slice.
TBUCFG_SI_R_HNDSHK_MODE	
TBUCFG_SI_AW_HNDSHK_MODE	
TBUCFG_SI_W_HNDSHK_MODE	
TBUCFG_SI_B_HNDSHK_MODE	
TBUCFG_MI_AR_HNDSHK_MODE	
TBUCFG_MI_R_HNDSHK_MODE	
TBUCFG_MI_AW_HNDSHK_MODE	
TBUCFG_MI_W_HNDSHK_MODE	
TBUCFG_MI_B_HNDSHK_MODE	

2.5.6 LTI TBU configuration options

You can configure the *Local Translation Interface* (LTI) TBU.

The following table shows the LTI TBU configuration options.

Table 2-30 LTI TBU configuration options

Configuration name	Options	Description
TBUCFG_LTI_ID_WIDTH	1-32	LTI ID width.
TBUCFG_LTI_LOOP_WIDTH	1-256	LTI loop width.

2.5.7 Common LTI TBU and ACE-Lite TBU configuration options

You can configure the common LTI TBU and ACE-Lite TBU options.

Common LTI TBU and ACE-Lite TBU configuration options

The following table shows the common LTI TBU and ACE-Lite TBU configuration options.

Table 2-31 Common LTI TBU and ACE-Lite TBU configuration options

Configuration name	Options	Description
TBUCFG_SID_WIDTH	8, 16, 20	Stream ID width.
TBUCFG_SSID_WIDTH	1, 8, 20	SubstreamID width.
TBUCFG_DIRECT_IDX	0, 1	Direct indexing. ——— **Note** ——— Must be 0 if TBUCFG_MTLB_DEPTH = 0. ———

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-31 Common LTI TBU and ACE-Lite TBU configuration options (continued)

Configuration name	Options	Description
TBUCFG_MTLB_PARTS	1, 2, 4, 8, 16	Number of main TLB partitions. **Note** Must be 1 if TBUCFG_MTLB_DEPTH = 0. Must be 1 if TBUCFG_DIRECT_IDX = 1. TBUCFG_MTLB_PARTS × TBUCFG_MTLB_DEPTH must not exceed 65536.
TBUCFG_LTI_OG_WIDTH	1-5	LTI ordering groups width. Number of ordering groups = 2^ TBUCFG_LTI_OG_WIDTH.
TBUCFG_LA_HNDSHK_MODE	0, 1, 2, 3	Handshake mode on Address channel before TLB lookup. Supported values are as follows: 0 FULL: Fully registered, double-buffered register slice. 1 FWD: Registered on the forward path only, that is, the direction that **xVALID** on the corresponding interface indicates. 2 REV: Registered on the reverse path only, that is, the direction that **xREADY** on the corresponding interface indicates. 3 BP: Bypass register slice.
TBUCFG_LR_HNDSHK_MODE	0, 1, 2, 3	Handshake mode on translation response path. Supported values are as follows: 0 or 1 FWD: Registered on the forward path only, that is, the direction that **LAVALID** on the corresponding interface indicates. 2 or 3 BP: Bypass register slice. **Note** If the TBUCFG_LR_HNDSHK_MODE parameter is set to: 0 or 1 The LTI master must provide at least three LR credits to achieve full utilization of the LTI interface. 2 or 3 The LTI master must provide at least two LR credits to achieve full utilization of the LTI interface.

2.5.8 TBU buffer configuration options

You can configure the TBU buffer.

The following table shows the TBU buffer configuration options.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 2-32 TBU buffer configuration options

Configuration name	Options	Description
TBUCFG_XLATE_SLOTS	2, 4, 8, 16, 32, 64, 128, 256, 512, 1024	Number of translation slots, controlling the Hit-Under-Miss capability of the TBU
TBUCFG_MTLB_LKP_SLOTS	2-28	Number of MTLB lookup slots. Use the default value to provide the best performance when one page size is active in the MTLB. You can: • Increase the value of this parameter if more than one page size is active in the MTLB • Decrease the value of this parameter if the TBU performance is not critical Ensure that the value of TBUCFG_MTLB_LKP_SLOTS is not greater than TBUCFG_XLATE_SLOTS
TBUCFG_UTLB_DEPTH	4, 8, 12, 16, 32, 64	Micro TLB depth, in entries
TBUCFG_MTLB_DEPTH	0, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536	Main TLB depth, in entries
TBUCFG_MTLB_WAYS	4, 8, 16	Number of ways in the MTLB
TBUCFG_MTLB_BANKS	1, 2, 4	Number of banks in the MTLB
TBUCFG_PMU_COUNTERS	4, 16, 32	Number of PMU counters
TBUCFG_PARTID_WIDTH	1, 6, 9	Width of PARTID supported. When set to 1, the **PARTID** that is received on the DTI interface is ignored because there is only one partition. When set to 6, **PARTID[8:6]** received on the DTI interface is ignored.
TBUCFG_HZRD_ENTRIES	0, 4, 8, 16, 32, 64	Number of hazard entries
TBUCFG_SLOTRAM_TYPE	0, 1, 2	RAM type for slot group of RAMs. 0 Two ports, that is, one port for reads and one port for writes. 1 One port, that is, one port for both reads and writes. 2 2 × one port, that is, banked configuration. See the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*. **Note** If this parameter is set to 2 and the depth of any particular RAM is 1 or 2, then the type is automatically set to 0. Arm recommends that the RAM is implemented as registers in these cases.
TBUCFG_CACHERAM_TYPE	0, 1	RAM type for cache group of RAMs. 0 Two ports, that is, one port for reads and one port for writes 1 One port, that is, one port for both reads and writes

2.5.9 TBU debug configuration options

You can configure the TBU debug options.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The following table shows the TBU debug configuration options.

Table 2-33 TBU debug configuration options

Configuration name	Options	Description
TBUCFG_USE_ELA_DEBUG	0, 1	Set the TBUCFG_USE_ELA_DEBUG parameter as follows: 0 The **SIGCLKEN<n>**, **SIGNALGRP<n>**, and **SIGQUAL<n>** signals are driven to 0 1 The **SIGCLKEN<n>**, **SIGNALGRP<n>**, and **SIGQUAL<n>** signals are driven according to: • *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209 • *B.3 LTI TBU observation interfaces* on page Appx-B-213

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2.6 Debug capability

The CoreLink MMU-700 System Memory Management Unit provides debug functionality using the CoreSight ELA-600 Embedded Logic Analyzer.

For more information about debug capability, see the *Arm® CoreLink™ MMU-700 System Memory Management Unit Configuration and Integration Manual*.

Note

The CoreSight ELA-600 Embedded Logic Analyzer is a separate licensed product that is not included with the CoreLink MMU-700 System Memory Management Unit.

Configuration options

For TCU configuration options, see *2.5.3 TCU debug configuration options* on page 2-78.

For TBU configuration options, see *2.5.9 TBU debug configuration options* on page 2-82.

Signals

For TCU observation signals, see *B.1 TCU observation interfaces* on page Appx-B-204.

For ACE-Lite TBU observation signals, see *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209.

For LTI TBU observation signals, see *B.3 LTI TBU observation interfaces* on page Appx-B-213.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Chapter 3
Programmers model

This chapter describes the MMU-700 programmers model.

It contains the following sections:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.1 About the programmers model

This section provides general information about the MMU-700 register properties.

The following information applies to the MMU-700 registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Access type is described as follows:

  RW Read and write

  RO Read-only

  WO Write-only

  RAZ Read-As-Zero

  WI Writes ignored
- Do not attempt to access reserved or unused address locations. Reading these locations results in RAZ and writing to these locations results in WI.
- Unless otherwise stated in the accompanying text:
  - Do-Not-Modify UNDEFINED register bits
  - Ignore UNDEFINED register bits on reads
  - All register bits are reset to 0 by a system or Cold reset
- Bit positions that are described as reserved are:
  - In an RW register, RAZ/WI
  - In an RO register, RAZ
  - In a WO register, WI

The MMU-700 registers are accessed using the PROG APB4 slave interface on the TCU, and cannot be accessed directly through any other slave interfaces.

Some registers are 64 bits, but the PROG APB4 interface is 32 bits. Because software accesses 64-bit registers 32 bits at a time, such accesses are not guaranteed to be 64-bit atomic. This behavior does not cause problems for software, because the SMMUv3 architecture does not require 64-bit atomic access to any registers.

The programmer's model contains separate TBU and TCU regions for internal control, RAS, and identification registers. Writes to unmapped or reserved registers are ignored, and reads SBZ. Non-secure accesses to Secure registers are RAZ/WI. The MMU-700 implements the identification register scheme that the SMMUv3 architecture defines.

The MMU-700 implements all the *Performance Monitor Counter Group* (PMCG) registers that the SMMUv3 architecture defines, except for:

- SMMU_PMCG_IRQ_CFG0
- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2

The MMU-700 does not implement the following SMMUv3 architectural registers, and accesses to these locations are RAZ/WI:

- SMMU_IDR4
- SMMU_STATUSR
- SMMU_GATOS_*
- SMMU_S_GATOS_*
- SMMU_VATOS_*

For more information about the SMMU architectural registers, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

This section contains the following subsection:

- *3.1.1 Clearing ERRSTATUS registers* on page 3-87.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.1.1 Clearing ERRSTATUS registers

Software can clear the TCU_ERRSTATUS and TBU_ERRSTATUS registers by writing ones to fields that are set.

For more information about these registers, see the following:

- *3.8.3 TCU_ERRSTATUS register* on page 3-116
- *3.15.3 TBU_ERRSTATUS register* on page 3-149

If both of the following are true, a write to the register is ignored:

- Any of the V, UE, OF, CE, DE, fields are nonzero before the write.
- The write does not clear the nonzero V, UE, OF, CE, DE fields to zero by writing ones to the applicable field or fields.

  Note

  CE must be cleared by writing `2'b11` to the field.

If a valid clearing write reaches the ERRSTATUS register on the same cycle as a new error, the new record is applied as though no previous error existed:

ERRSTATUS.V = 0.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.2 SMMU architectural registers

The MMU-700 implements many of the SMMU architectural registers, that the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

The following table lists the SMMUv3 architectural registers that the MMU-700 implements.

Note

All writable register fields reset to 0 unless the SMMU architecture specifies otherwise.

Table 3-1 SMMUv3 architectural registers

Register	Name	Description
SMMU_S_IDR0 - SMMU_S_IDR3	SMMU Secure feature Identification Registers	Provides information about the Secure features that the SMMU implementation supports
SMMU_S_CR0	Secure global Control Register 0	Provides global configuration of the Secure SMMU
SMMU_S_CR0ACK	Secure global Control Register 0 update Acknowledge	Provides acknowledgment of completion of updates to SMMU_S_CR0
SMMU_S_CR1 SMMU_S_CR2	Secure global Control Registers	Provides the controls for Secure table and queue access attributes
SMMU_S_INIT	Secure Initialization control register	Provides a control to invalidate all Secure SMMU caching on system initialization
SMMU_S_GBPA	Secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Secure streams when the MMU is disabled
SMMU_S_IRQ_CTRL	Secure Interrupt Control register	Contains enables for SMMU interrupts
SMMU_S_IRQ_CTRLACK	Secure Interrupt Control register update Acknowledge	Provides acknowledgment of the completion of updates to SMMU_S_IRQ_CTRL
SMMU_S_GERROR	Secure Global Error status register	Provides information on Secure global programming interface errors
SMMU_S_GERRORN	Secure Global Error Acknowledgment register	Contains the acknowledgment fields for SMMU_S_GERROR errors
SMMU_S_GERROR_IRQ_CFG0 - SMMU_S_GERROR_IRQ_CFG2	Secure Global Error IRQ Configuration register	Contains the Secure MSI address configuration for the GERROR IRQ
SMMU_S_STRTAB_BASE	Secure Stream Table Base address register	Contains the base address and attributes for the Secure Stream table
SMMU_S_STRTAB_BASE_CFG	Secure Stream Table Base Configuration register	Contains configuration fields for the Secure Stream table
SMMU_S_CMDQ_BASE	Secure Command queue Base address register	Contains the base address and attributes for the Secure Command queue

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-1 SMMUv3 architectural registers (continued)

Register	Name	Description
SMMU_S_CMDQ_PROD	Secure Command queue Producer index register	Contains the Secure Command queue index for writes by the producer
SMMU_S_CMDQ_CONS	Secure Command queue Consumer index register	Contains the Secure Command queue index for reads by the consumer
SMMU_S_EVENTQ_BASE	Secure Event queue Base address register	Contains the base address and attributes for the Secure Event queue
SMMU_S_EVENTQ_PROD	Secure Event queue Producer index register	Contains the Secure Event queue index for writes by the producer
SMMU_S_EVENTQ_CONS	Secure Event queue Consumer index register	Contains the Secure Event queue index for reads by the consumer
SMMU_S_EVENTQ_IRQ_CFG0 - SMMU_S_EVENTQ_IRQ_CFG2	Secure Event queue IRQ Configuration registers	Contains the MSI address configuration for the Secure Event queue IRQ
SMMU_IDR0 - SMMU_IDR3 SMMU_IDR5	SMMU feature Identification Registers	Provides information about the features that the SMMU implementation supports
SMMU_IIDR	Implementation Identification Register	Provides implementer, part, and revision information for the SMMU implementation
SMMU_AIDR	Architecture Identification Register	Identifies the SMMU architecture version to which the implementation conforms
SMMU_CR0	Non-secure global Control Register 0	Provides the controls for the global configuration of the Non-secure SMMU
SMMU_CR0ACK	Non-secure global Control Register 0 update Acknowledge register	Provides acknowledgment of completion of updates to SMMU_CR0
SMMU_CR1	Non-secure global Control Register 1	Provides the controls for Non-secure table and queue access attributes
SMMU_CR2	Non-secure global Control Register 2	Provides the controls for the configuration of the global Non-secure features
SMMU_GBPA	Non-secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Non-secure streams when the MMU is disabled
SMMU_IRQ_CTRL	Non-secure Interrupt Control register	Provides IRQ enable flags for edge-triggered wired outputs, if implemented, and MSI writes, if implemented
SMMU_IRQ_CTRLACK	Non-secure Interrupt Control register update Acknowledge register	Provides acknowledgment of the completion of updates to SMMU_IRQ_CTRL
SMMU_GERROR	Non-secure Global Error status register	Provides information about Non-secure global programming interface errors

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-1 SMMUv3 architectural registers (continued)

Register	Name	Description
SMMU_GERRORN	Non-secure Global Error acknowledgment register	Contains the acknowledgment fields for SMMU_GERROR errors
SMMU_GERROR_IRQ_CFG0	Non-secure Global Error IRQ Configuration register 0	Contains the MSI address configuration for the GERROR IRQ
SMMU_GERROR_IRQ_CFG1	Non-secure Global Error IRQ Configuration register 1	Contains the MSI payload configuration for the GERROR IRQ
SMMU_GERROR_IRQ_CFG2	Non-secure Global Error IRQ Configuration register 2	Contains the MSI attribute configuration for the GERROR IRQ
SMMU_STRTAB_BASE	Non-secure Stream Table Base address register	Contains the base address and attributes for the Non-secure Stream table
SMMU_STRTAB_BASE_CFG	Non-secure Stream Table Configuration register	Contains configuration fields for the Non-secure Stream table
SMMU_CMDQ_BASE	Non-secure Command queue Base address register	Contains the base address and attributes for the Non-secure Command queue
SMMU_CMDQ_PROD	Non-secure Command queue Producer index register	Contains the Non-secure Command queue index for writes by the producer
SMMU_CMDQ_CONS	Non-secure Command queue Consumer index register	Contains the Non-secure Command queue index for reads by the consumer
SMMU_EVENTQ_BASE	Non-secure Event queue Base address register	Contains the base address and attributes for the Non-secure Event queue
SMMU_EVENTQ_PROD	Non-secure Event queue Producer index register	Contains the Non-secure Event queue index for writes by the producer
SMMU_EVENTQ_CONS	Non-secure Event queue Consumer index register	Contains the Non-secure Event queue index for reads by the consumer
SMMU_EVENTQ_IRQ_CFG0	Non-secure Event queue IRQ Configuration register 0	Contains the MSI address configuration for the Event queue IRQ
SMMU_EVENTQ_IRQ_CFG1	Non-secure Event queue IRQ Configuration register 1	Contains the MSI payload configuration for the Event queue IRQ
SMMU_EVENTQ_IRQ_CFG2	Non-secure Event queue IRQ Configuration register 2	Contains the MSI attribute configuration for the Event queue IRQ
SMMU_PRIQ_BASE	Non-secure PRI queue Base address register	Contains the base address and attributes for the Non-secure PRI queue
SMMU_PRIQ_PROD	Non-secure PRI queue Producer index register	Contains the Non-secure PRI queue index for writes by the producer
SMMU_PRIQ_CONS	Non-secure PRI queue Consumer index register	Contains the Non-secure PRI queue index for reads by the consumer

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-1 SMMUv3 architectural registers (continued)

Register	Name	Description
SMMU_PRIQ_IRQ_CFG0	Non-secure PRI queue IRQ Configuration register 0	Contains the MSI address configuration for the PRI queue IRQ
SMMU_PRIQ_IRQ_CFG1	Non-secure PRI queue IRQ Configuration register 1	Contains the MSI payload configuration for the PRI queue IRQ
SMMU_PRIQ_IRQ_CFG2	Non-secure PRI queue IRQ Configuration register 2	Contains the MSI attribute configuration for the PRI queue IRQ

The MMU-700 implements an SMMUv3 *Performance Monitor Counter Group* (PMCG) in the TCU and in each TBU. The following table lists the registers that the MMU-700 implements in each PMCG.

Table 3-2 SMMUv3 PMCG registers

Register	Name	Description
SMMU_PMCG_EVCNTR0 - SMMU_PMCG_EVCNTR3	SMMU PMCG Event Counter registers	Contains the values of the event counters
SMMU_PMCG_EVTYPER0 - SMMU_PMCG_EVTYPER3	SMMU PMCG Event Type configuration registers	Configures the events that the corresponding counter counts
SMMU_PMCG_SVR0 - SMMU_PMCG_SVR3	SMMU PMCG Shadow Value Registers	Contains the shadow value of the corresponding event counter
SMMU_PMCG_SMR0	SMMU PMCG Stream Match filter Register	Configures the stream match filter for the corresponding event counter
SMMU_PMCG_CNTENSET0	SMMU PMCG Counter Enable Set register	Provides the set mechanism for the counter enables
SMMU_PMCG_CNTENCLR0	SMMU PMCG Counter Enable Clear register	Provides the clear mechanism for the counter enables
SMMU_PMCG_INTENSET0	SMMU PMCG Interrupt contribution Enable Set register	Provides the set mechanism for the counter interrupt contribution enables
SMMU_PMCG_INTENCLR0	SMMU PMCG Interrupt contribution Enable Clear register	Provides the clear mechanism for the counter interrupt enables
SMMU_PMCG_OVSCLR0	SMMU PMCG Overflow Status Clear register	Provides the clear mechanism for the overflow status bits and provides read access to the overflow status bit values
SMMU_PMCG_OVSSET0	SMMU PMCG Overflow Status Set register	Provides the set mechanism for the overflow status bits and provides read access to the overflow status bit values
SMMU_PMCG_CAPR	SMMU PMCG Counter shadow value Capture Register	Controls the counter shadow value capture mechanism

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-2 SMMUv3 PMCG registers (continued)

Register	Name	Description
SMMU_PMCG_SCR	SMMU PMCG Secure Control Register	Secure Control Register
SMMU_PMCG_CFGR	SMMU PMCG Configuration information Register	Provides information about the PMCG implementation
SMMU_PMCG_CR	SMMU PMCG Control Register	Contains the Performance Monitor control flags
SMMU_PMCG_CEID0 - SMMU_PMCG_CEID1	SMMU PMCG Common Event ID registers	Contains the lower and upper 64 bits of the Common Event identification bitmap
SMMU_PMCG_IRQ_CTRL	SMMU PMCG IRQ enable register	Contains the Performance Monitors IRQ enable
SMMU_PMCG_IRQ_CTRLACK	SMMU PMCG IRQ enable Acknowledge register	Provides acknowledgment of the completion of updates to SMMU_PMCG_IRQ_CTRL
SMMU_PMCG_AIDR	SMMU PMCG Architecture Identification Register	Provides the Performance Monitor Architecture Identification
SMMU_PMCG_ID_REGS	ID registers	IMPLEMENTATION DEFINED
SMMU_PMCG_PMAUTHSTATUS	PMU Authentication Status register	Performance Monitor authentication status
SMMU_PMCG_PMDEVARCH	PMU Device Architecture register	Performance Monitor architecture identifier
SMMU_PMCG_PMDEVTYPE	PMU Device Type register	Performance Monitor device type

Related concepts

2.4.1 SMMUv3 implementation on page 2-57

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.3 MMU-700 memory map

The MMU-700 memory map contains all registers.

This section contains the following subsections:

- *3.3.1 Main MMU-700 memory map* on page 3-93.
- *3.3.2 TCU memory map* on page 3-93.
- *3.3.3 TBU memory map* on page 3-94.

3.3.1 Main MMU-700 memory map

The main MMU-700 memory map includes the TCU and all TBUs, and the maximum number of implemented TBUs.

The following table shows the full memory map.

Table 3-3 Main MMU-700 memory map

Address range	Description
`0x000000` - `0x03FFFC`	TCU registers
`0x040000` - `0x05FFFC`	TBU0 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
`0x060000` - `0x07FFFC`	TBU1 registers. Include microarchitectural, RAS, ID, MPAM, and PMCG registers.
`0x080000` - `0x09FFFC`	TBU2 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
...	...
`0x7C0000` - `0x7DFFFC`	TBU60 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.
`0x7E0000` - `0x7FFFFC`	TBU61 registers. Includes microarchitectural, RAS, ID, MPAM, and PMCG registers.

Note

This document describes all TBU and TCU register addresses relative to the base address for that component.

3.3.2 TCU memory map

The TCU memory map contains various categories of registers.

The TCU IMPLEMENTATION DEFINED registers include the following:

- *3.7 TCU microarchitectural registers* on page 3-106 for controlling microarchitectural features
- *3.9 TCU system discovery registers* on page 3-120
- *3.8 TCU RAS registers* on page 3-114
- *3.6 TCU PMU registers* on page 3-103

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

The following registers are also included:

- *3.10 TCU PIU integration registers* on page 3-135.
- Walk cache stage and level *Memory System Resource Partitioning and Monitoring* (MPAM) maximum capacity registers.
- MPAM memory-mapped registers.

The following table shows the MMU-700 TCU memory map.

Table 3-4 MMU-700 TCU memory map

Address range	Description
0x00000-0x0FFFC	TCU registers, page 0, including: • SMMUv3 registers, page 0 • TCU *Performance Monitor Counter Group* (PMCG) registers, page 0, starting at offset 0x02000 • TCU microarchitectural registers • TCU system discovery registers • TCU MPAM registers
0x10000-0x1FFFC	TCU registers, page 1. This address range contains the SMMUv3 registers, page 1.
0x20000-0x2FFFC	TCU registers, page 2. This address range contains the TCU PMCG registers, page 1, starting at offset 0x22000.
0x30000-0x3FFFC	Reserved.

The following table shows how the TCU IMPLEMENTATION DEFINED PMCG, and MPAM registers are allocated to regions of the TCU address space. Other regions are reserved.

Table 3-5 TCU PMCG, RAS, and MPAM register allocation to regions of TCU address space

Address range	Description
0x00FD0-0x00FFC	SMMU ID registers
0x02000-0x02FFC	Performance Monitor, page 0
0x03000-0x03FFC	MPAM Non-secure registers
0x08E00-0x08E7C	Microarchitectural features and integration registers
0x08E34-0x08E78	System discovery registers
0x08E80-0x08EFC	*Reliability, Availability, and Serviceability* (RAS) registers
0x09000-0x097FC	TCU node microarchitecture registers
0x09800-0x0981C	Walk cache stage and level MPAM maximum capacity registers
0x0B000-0x0BFFC	MPAM Secure registers
0x22000-0x22FFC	Performance Monitor, page 1

3.3.3 TBU memory map

The TBU memory map contains various categories of registers.

The TBU registers contain the following:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

- IMPLEMENTATION DEFINED *3.14 TBU microarchitectural registers* on page 3-144 for controlling microarchitectural features
- *3.16 TBU system discovery registers* on page 3-153
- *3.15 TBU RAS registers* on page 3-147
- Direct access to cache state
- *3.13 TBU PMU registers* on page 3-141
- Performance Monitor counter registers, on a separate 64KB page to enable it to be paged for direct access from a Guest OS

The following table shows the TBU memory map.

Table 3-6 TBU memory map

Address range	Description
`0x08E00-0x08E7C`	Microarchitectural registers. System discovery registers. Integration registers.
`0x08E80-0x08EFC`	RAS
`0x00FD0-0x00FFC`	ID registers
`0x02000-0x02FFC`	Performance Monitor page 0
`0x12000-0x12FFC`	Performance Monitor page 1
`0x03000-0x03FFC`	TBU MPAM Non-secure registers
`0x0B000-0x0BFFC`	TBU MPAM Secure registers

Note

Any regions that the table does not show are reserved.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.4 MMU-700 registers summary

The register summary describes the MMU-700 registers and some key characteristics.

This section contains the following subsections:

3.4.1 TCU identification registers summary

The MMU-700 contains TCU identification registers.

The following table shows the TCU identification registers in offset order from the base memory address.

Table 3-7 TCU identification registers summary

Name	Offset	Type	Description
SMMU_CIDR3	0x00FFC	RO	*3.5 TCU component and peripheral ID registers* on page 3-102
SMMU_CIDR2	0x00FF8	RO	
SMMU_CIDR1	0x00FF4	RO	
SMMU_CIDR0	0x00FF0	RO	
SMMU_PIDR3	0x00FEC	RO	
SMMU_PIDR2	0x00FE8	RO	
SMMU_PIDR1	0x00FE4	RO	
SMMU_PIDR0	0x00FE0	RO	
SMMU_PIDR7	0x00FDC	RO	
SMMU_PIDR6	0x00FD8	RO	
SMMU_PIDR5	0x00FD4	RO	
SMMU_PIDR4	0x00FD0	RO	

3.4.2 TCU and TBU PMU identification registers summary

The TCU and the TBU use the same PMU identification registers.

The following table shows the TCU and TBU PMU identification registers in offset order from the base memory address.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-8 TCU and TBU PMU identification registers summary

Name	Offset	Type	Description
SMMU_PMCG_PMAUTHSTATUS	0x00FB8	RO	*3.6 TCU PMU registers* on page 3-103 *3.13 TBU PMU registers* on page 3-141
SMMU_PMCG_PIDR4	0x02FD0	RO	
SMMU_PMCG_PIDR5	0x02FD4	RO	
SMMU_PMCG_PIDR6	0x02FD8	RO	
SMMU_PMCG_PIDR7	0x02FDC	RO	
SMMU_PMCG_PIDR0	0x02FE0	RO	
SMMU_PMCG_PIDR1	0x02FE4	RO	
SMMU_PMCG_PIDR2	0x02FE8	RO	
SMMU_PMCG_PIDR3	0x02FEC	RO	
SMMU_PMCG_CIDR0	0x02FF0	RO	
SMMU_PMCG_CIDR1	0x02FF4	RO	
SMMU_PMCG_CIDR2	0x02FF8	RO	
SMMU_PMCG_CIDR3	0x02FFC	RO	

3.4.3 TCU Reliability, Availability, and Service registers summary

The MMU-700 contains TCU *Reliability, Availability, and Service* (RAS) registers.

The following table shows the TCU RAS registers in offset order from the base memory address.

Table 3-9 TCU RAS registers summary

Name	Offset	Type	Width	Description
TCU_ERRFR	0x08E80	RO, Secure	64-bit	*3.8.1 TCU_ERRFR register* on page 3-114
TCU_ERRCTLR	0x08E88	RW, Secure	64-bit	*3.8.2 TCU_ERRCTLR register* on page 3-115
TCU_ERRSTATUS	0x08E90	RW, Secure	64-bit	*3.8.3 TCU_ERRSTATUS register* on page 3-116
TCU_ERRGEN	0x08EC0	RW, Secure	64-bit	*3.8.4 TCU_ERRGEN register* on page 3-119

3.4.4 TCU microarchitectural registers summary

The MMU-700 contains TCU microarchitectural registers.

The following table shows the TCU microarchitectural registers in offset order from the base memory address.

Table 3-10 TCU microarchitectural registers summary

Name	Offset	Type	Width	Description
TCU_CTRL	0x08E00	RW	32-bit	*3.7.1 TCU_CTRL register* on page 3-106
TCU_QOS	0x08E04	RW	32-bit	*3.7.2 TCU_QOS register* on page 3-107
TCU_CFG	0x08E08	RO	32-bit	*3.7.3 TCU_CFG register* on page 3-108
TCU_STATUS	0x08E10	RO	32-bit	*3.7.4 TCU_STATUS register* on page 3-109
TCU_SCR	0x08E18	RW, Secure	32-bit	*3.7.7 TCU_SCR register* on page 3-112

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-10 TCU microarchitectural registers summary (continued)

Name	Offset	Type	Width	Description
TCU_NODE_CTRL*n*	0x09000-0x093FC	RW	32-bit	*3.7.5 TCU_NODE_CTRLn register* on page 3-110
TCU_NODE_STATUS*n*	0x09400-0x097FC	RO	32-bit	*3.7.6 TCU_NODE_STATUSn register* on page 3-111
TCU_WC_SxLy_CMAX	0x09800-0x0981C	RW	32-bit	*3.7.8 TCU_WC_SxLy_CMAX registers* on page 3-113

3.4.5 TCU system discovery registers summary

The MMU-700 contains TCU system discovery registers.

The following table shows the TCU system discovery registers in offset order from the base memory address.

Table 3-11 TCU system discovery registers summary

Name	Offset	Type	Width	Description
TCU_SYSDISC0	0x08E34	RO	32-bit	*3.9.1 TCU_SYSDISC0 system discovery register* on page 3-120
TCU_SYSDISC1	0x08E38	RO	32-bit	*3.9.2 TCU_SYSDISC1 system discovery register* on page 3-121
TCU_SYSDISC2	0x08E3C	RO	32-bit	*3.9.3 TCU_SYSDISC2 system discovery register* on page 3-122
TCU_SYSDISC3	0x08E40	RO	32-bit	*3.9.4 TCU_SYSDISC3 system discovery register* on page 3-122
TCU_SYSDISC4	0x08E44	RO	32-bit	*3.9.5 TCU_SYSDISC4 system discovery register* on page 3-123
TCU_SYSDISC5	0x08E48	RO	32-bit	*3.9.6 TCU_SYSDISC5 system discovery register* on page 3-124
TCU_SYSDISC6	0x08E4C	RO	32-bit	*3.9.7 TCU_SYSDISC6 system discovery register* on page 3-125
TCU_SYSDISC7	0x08E50	RO	32-bit	*3.9.8 TCU_SYSDISC7 system discovery register* on page 3-126
TCU_SYSDISC8	0x08E54	RO	32-bit	*3.9.9 TCU_SYSDISC8 system discovery register* on page 3-126
TCU_SYSDISC9	0x08E58	RO	32-bit	*3.9.10 TCU_SYSDISC9 system discovery register* on page 3-127
TCU_SYSDISC10	0x08E5C	RO	32-bit	*3.9.11 TCU_SYSDISC10 system discovery register* on page 3-128
TCU_SYSDISC11	0x08E60	RO	32-bit	*3.9.12 TCU_SYSDISC11 system discovery register* on page 3-129
TCU_SYSDISC12	0x08E64	RO	32-bit	*3.9.13 TCU_SYSDISC12 system discovery register* on page 3-130
TCU_SYSDISC13	0x08E68	RO	32-bit	*3.9.14 TCU_SYSDISC13 system discovery register* on page 3-130
TCU_SYSDISC14	0x08E6C	RO	32-bit	*3.9.15 TCU_SYSDISC14 system discovery register* on page 3-131
TCU_SYSDISC15	0x08E70	RO	32-bit	*3.9.16 TCU_SYSDISC15 system discovery register* on page 3-132
TCU_SYSDISC16	0x08E74	RO	32-bit	*3.9.17 TCU_SYSDISC16 system discovery register* on page 3-133
TCU_SYSDISC17	0x08E78	RO	32-bit	*3.9.18 TCU_SYSDISC17 system discovery register* on page 3-134

3.4.6 TCU integration registers summary

The MMU-700 contains TCU integration registers.

The following table shows the TCU integration registers in offset order from the base memory address.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-12 TCU integration registers summary

Name	Offset	Type	Width	Description
ITEN	0x08E20	RW	32-bit	*3.10.1 ITEN register for the TCU* on page 3-135
ITOP_PIU	0x08E24	RW	32-bit	*3.10.2 ITOP register for the TCU Programmer Interface Unit* on page 3-135
ITOP_TMU	0x08E2C	RW	32-bit	*3.11.1 ITOP register for the TCU Translation Management Unit* on page 3-138
ITIN_TMU	0x08E30	RO	32-bit	*3.11.2 ITIN register for the TCU Translation Management Unit* on page 3-139

3.4.7 TBU identification registers summary

The MMU-700 contains TBU identification registers.

The following table shows the TBU identification registers in offset order from the base memory address.

Table 3-13 TBU identification registers summary

Name	Offset	Type	Description
SMMU_CIDR3	0x00FFC	RO	*3.12 TBU component and peripheral ID registers* on page 3-140
SMMU_CIDR2	0x00FF8	RO	
SMMU_CIDR1	0x00FF4	RO	
SMMU_CIDR0	0x00FF0	RO	
SMMU_PIDR3	0x00FEC	RO	
SMMU_PIDR2	0x00FE8	RO	
SMMU_PIDR1	0x00FE4	RO	
SMMU_PIDR0	0x00FE0	RO	
SMMU_PIDR7	0x00FDC	RO	
SMMU_PIDR6	0x00FD8	RO	
SMMU_PIDR5	0x00FD4	RO	
SMMU_PIDR4	0x00FD0	RO	

3.4.8 TBU Reliability, Availability, and Serviceability registers summary

The MMU-700 contains TBU *Reliability, Availability, and Serviceability* (RAS) registers.

The following table shows the TBU RAS registers in offset order from the base memory address.

Table 3-14 TBU RAS registers summary

Name	Offset	Width	Type	Description
TBU_ERRFR	0x08E80	64-bit	RO, Secure	*3.15.1 TBU_ERRFR register* on page 3-147
TBU_ERRCTLR	0x08E88	64-bit	RW, Secure	*3.15.2 TBU_ERRCTLR register* on page 3-148
TBU_ERRSTATUS	0x08E90	64-bit	RW, Secure	*3.15.3 TBU_ERRSTATUS register* on page 3-149
TBU_ERRGEN	0x08EC0	64-bit	RW, Secure	*3.15.4 TBU_ERRGEN register* on page 3-152

RAS error reporting

When a *Correctable Error* (CE) occurs:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A CE is reported in *3.15.3 TBU_ERRSTATUS register* on page 3-149.

If TBU_ERRCTLR.FI is set, an interrupt is raised on **ras_fhi**. See *TBU interrupt interfaces* on page 2-37.

3.4.9 TBU microarchitectural registers summary

The MMU-700 contains TBU microarchitectural registers.

The following table shows the TBU microarchitectural registers in offset order from the base memory address.

Table 3-15 TBU microarchitectural registers summary

Name	Offset	Type	Width	Description
TBU_CTRL	0x08E00	RW	32-bit	*3.14.1 TBU_CTRL register* on page 3-144
TBU_SCR	0x08E18	RW, Secure	32-bit	*3.14.2 TBU_SCR register* on page 3-145

3.4.10 TBU system discovery registers summary

The MMU-700 contains TBU system discovery registers.

The following table shows the TBU system discovery registers in offset order from the base memory address.

Table 3-16 TBU system discovery registers summary

Name	Offset	Type	Width	Description
TBU_SYSDISC0	0x08E30	RO	32-bit	*3.16.1 TBU_SYSDISC0 system discovery register* on page 3-153
TBU_SYSDISC1	0x08E34	RO	32-bit	*3.16.2 TBU_SYSDISC1 system discovery register* on page 3-154
TBU_SYSDISC2	0x08E38	RO	32-bit	*3.16.3 TBU_SYSDISC2 system discovery register* on page 3-155
TBU_SYSDISC3	0x08E3C	RO	32-bit	*3.16.4 TBU_SYSDISC3 system discovery register* on page 3-155
TBU_SYSDISC4	0x08E40	RO	32-bit	*3.16.5 TBU_SYSDISC4 system discovery register* on page 3-156
TBU_SYSDISC5	0x08E44	RO	32-bit	*3.16.6 TBU_SYSDISC5 system discovery register* on page 3-157
TBU_SYSDISC6	0x08E48	RO	32-bit	*3.16.7 TBU_SYSDISC6 system discovery register* on page 3-158
TBU_SYSDISC7	0x08E4C	RO	32-bit	*3.16.8 TBU_SYSDISC7 system discovery register* on page 3-159
TBU_SYSDISC8	0x08E50	RO	32-bit	*3.16.9 TBU_SYSDISC8 system discovery register* on page 3-159
TBU_SYSDISC9	0x08E54	RO	32-bit	*3.16.10 TBU_SYSDISC9 system discovery register* on page 3-160
TBU_SYSDISC10	0x08E58	RO	32-bit	*3.16.11 TBU_SYSDISC10 system discovery register* on page 3-161
TBU_SYSDISC11	0x08E5C	RO	32-bit	*3.16.12 TBU_SYSDISC11 system discovery register* on page 3-162
TBU_SYSDISC12	0x08E60	RO	32-bit	*3.16.13 TBU_SYSDISC12 system discovery register* on page 3-163
TBU_SYSDISC13	0x08E64	RO	32-bit	*3.16.14 TBU_SYSDISC13 system discovery register* on page 3-163
TBU_SYSDISC14	0x08E68	RO	32-bit	*3.16.15 TBU_SYSDISC14 system discovery register* on page 3-164

3.4.11 TBU integration registers summary

The MMU-700 contains TBU integration registers.

The following table shows the TBU integration registers in offset order from the base memory address.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-17 TBU integration registers summary

Name	Offset	Type	Width	Description
ITEN	0x08E20	RW	32-bit	*3.17.1 ITEN register* on page 3-166
ITOP_TBU	0x08E24	RW	32-bit	*3.17.2 ITOP_TBU register* on page 3-166
ITIN_TBU	0x08E28	RW	32-bit	*3.17.3 ITIN_TBU register* on page 3-167

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.5 TCU component and peripheral ID registers

This section describes the TCU component and peripheral ID registers.

The following table shows the TCU component and peripheral ID registers.

Table 3-18 TCU component and peripheral ID registers

Name	Offset	Field	Value	Description
SMMU_CIDR3, Component ID3	0x00FFC	[7:0]	0xB1	Preamble
SMMU_CIDR2, Component ID2	0x00FF8	[7:0]	0x05	Preamble
SMMU_CIDR1, Component ID1	0x00FF4	[7:0]	0xF0	Preamble
SMMU_CIDR0, Component ID0	0x00FF0	[7:0]	0x0D	Preamble
SMMU_PIDR3, Peripheral ID3	0x00FEC	[7:4]	MAX(*p_level*, **ecorevnum**)	REVAND, minor revision Where: *p_level* is: 0 For p0 1 For p1
		[3:0]	0x00	CMOD
SMMU_PIDR2, Peripheral ID2	0x00FE8	[7:4]	0x00	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
SMMU_PIDR1, Peripheral ID1	0x00FE4	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PIDR0, Peripheral ID0	0x00FE0	[7:0]	0x87	PART_0: bits [7:0] of the Part number
SMMU_PIDR7, Peripheral ID7	0x00FDC	-	RES0	Reserved
SMMU_PIDR6, Peripheral ID6	0x00FD8			
SMMU_PIDR5, Peripheral ID5	0x00FD4			
SMMU_PIDR4, Peripheral ID4	0x00FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.6 TCU PMU registers

This section describes the *Performance Monitor Unit* (PMU) registers. The Performance Monitor counter registers, on a separate 64KB page, enable it to be paged for direct access from a Guest OS.

This section contains the following subsections:

3.6.1 Registers

The TBU and TCU support the same PMCG registers.

These registers follow the register layout that the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* Performance Monitor Extension describes.

The following PMCG registers, that the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines, are implemented:

- SMMU_PMCG_EVCNTR{0-(TCUCFG_PMU_COUNTERS-1)}
- SMMU_PMCG_EVTYPER{0-(TCUCFG_PMU_COUNTERS-1)}
- SMMU_PMCG_SVR{0-(TCUCFG_PMU_COUNTERS-1)}
- SMMU_PMCG_SMR0
  - All counters share this mask register
  - The mask is 24 bits because the TCU uses 24-bit StreamIDs
- SMMU_PMCG_CNTENSET0
- SMMU_PMCG_CNTENCLR0
- SMMU_PMCG_INTENSET0
- SMMU_PMCG_INTSENCLR0
- SMMU_PMCG_OVSCLR0
- SMMU_PMCG_OVSSET0
- SMMU_PMCG_CAPR
- SMMU_PMCG_SCR
- SMMU_PMCG_CFGR. See *3.6.3 SMMU_PMCG_CFGR* on page 3-104.
- SMMU_PMCG_CR
- SMMU_PMCG_CEID{0-1}. See *3.6.4 SMMU_PMCG_CEID{0-1} registers* on page 3-104.
- SMMU_PMCG_IRQ_CTRL
- SMMU_PMCG_IRQ_CTRLACK
- SMMU_PMCG_AIDR, indicates SMMUv3.2
- SMMU_PMCG_ID_REGS

The following registers are not implemented, because the PMCG does not support MSIs:

- SMMU_PMCG_IRQ_CFG0
- SMMU_PMCG_IRQ_CFG1
- SMMU_PMCG_IRQ_CFG2
- SMMU_PMCG_IRQ_STATUS

The following registers are not implemented, because the PMCG implementation does not support MPAM:

- SMMU_PMCG_GMPAM
- SMMU_PMCG_MPAMIDR
- SMMU_PMCG_S_MPAMIDR

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.6.2 Events

In this description, a translation request corresponds to a translation slot allocation.

A single DTI translation request might correspond to multiple translation request events if:

- A translation results in a stall fault event and is restarted
- A translation results in a stall fault event when the Event queue is full, and is later retried when the Event queue becomes non-full

Each event indicates:

- Whether the SMMU_PMCG_SMR0 register can filter it
- For events that cannot be filtered, whether they are only visible when Secure events are visible by SMMU_PMCG_SCR.SO = 1

For more information about the architectural and IMPLEMENTATION DEFINED events that are implemented, see *SMMUv3 architectural performance events* on page 2-41.

The following events are also counted for prefetch accesses:

0x80-0x90
Walk cache events.

0x92-0x94
Configuration cache events.

0xC0-0xC8
RAS events.

3.6.3 SMMU_PMCG_CFGR

An MMU-700 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

For information about the SMMU_PMCG_CFGR field values, see *SMMUv3 PMU register architectural options* on page 2-45.

3.6.4 SMMU_PMCG_CEID{0-1} registers

The SMMU_PMCG_CEID{0-1} registers indicate the architectural events that are supported. They are described as 64-bit registers, but are accessed 32 bits at a time through the 32-bit PROG interface.

The following table shows the SMMU_PMCG_CEID{0-1} registers.

Table 3-19 SMMU_PMCG_CEID{0-1} registers

Address	Register	Value
0x02E20	SMMU_PMCG_CEID0	0x0000007F
0x02E28	SMMU_PMCG_CEID1	0x00000000

3.6.5 PMU ID registers

The PMU ID registers appear only in Performance Monitor Page 0. Page 1 does not contain any ID registers.

The following table shows the PMU ID registers.

Table 3-20 PMU ID registers

Address	Name	Field	Value	Description
0x02FFC	SMMU_PMCG_CIDR3, Component ID3	[7:0]	0xB1	Preamble
0x02FF8	SMMU_PMCG_CIDR2, Component ID2	[7:0]	0x05	Preamble

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-20 PMU ID registers (continued)

Address	Name	Field	Value	Description
0x02FF4	SMMU_PMCG_CIDR1, Component ID1	[7:0]	0x90	Preamble
0x02FF0	SMMU_PMCG_CIDR0, Component ID0	[7:0]	0x0D	Preamble
0x02FEC	SMMU_PMCG_PIDR3, Peripheral ID3	[7:4]	MAX(*p_level*, **ecorevnum**)	REVAND, minor revision, where *p_level* is: 0 For p0 1 For p1
		[3:0]	0x00	CMOD
0x02FE8	SMMU_PMCG_PIDR2, Peripheral ID2	[7:4]	0x00	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
0x02FE4	SMMU_PMCG_PIDR1, Peripheral ID1	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
0x02FE0	SMMU_PMCG_PIDR0, Peripheral ID0	[7:0]	0x87	PART_0: bits [7:0] of the Part number
0x02FDC	SMMU_PMCG_PIDR7, Peripheral ID7	-	RES0	Reserved
0x02FD8	SMMU_PMCG_PIDR6, Peripheral ID6	-	RES0	Reserved
0x02FD4	SMMU_PMCG_PIDR5, Peripheral ID5	-	RES0	Reserved
0x02FD0	SMMU_PMCG_PIDR4, Peripheral ID4	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code
0x00FB8	SMMU_PMCG_PMAUTHSTATUS	[7:0]	0x00	No authentication interface is implemented

The PMDEVARCH and PMDEVTYPE registers are implemented as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.7 TCU microarchitectural registers

You can set the TCU microarchitectural registers at boot time to optimize TCU behavior for your system. Arm recommends that you use the default values for most systems.

The *3.7.7 TCU_SCR register* on page 3-112 is Secure-only. Non-secure access to this register is *Read-As-Zero* (RAZ)/*Write-Ignored* (WI).

TCU_SCR.NS_UARCH controls Non-secure access to registers in this section other than TCU_SCR. Non-secure accesses to these registers, when TCU_SCR.NS_UARCH = 0, are RAZ and WI.

The following registers:

- *3.7.1 TCU_CTRL register* on page 3-106
- *3.7.2 TCU_QOS register* on page 3-107
- *3.7.5 TCU_NODE_CTRLn register* on page 3-110
- *3.7.8 TCU_WC_SxLy_CMAX registers* on page 3-113

Can only be written when the following occur:

- SMMU_CR0.SMMUEN = 0.
- SMMU_CR0ACK.SMMUEN = 0.
- SMMU_S_CR0.SMMUEN = 0.
- SMMU_S_CR0ACK.SMMUEN = 0.

After modifying these registers, software must issue an INV_ALL operation using the SMMU_S_INIT register, before it sets SMMUEN to 1. Failure to issue the operation results in UNPREDICTABLE behavior.

This section contains the following subsections:

- *3.7.1 TCU_CTRL register* on page 3-106.
- *3.7.2 TCU_QOS register* on page 3-107.
- *3.7.3 TCU_CFG register* on page 3-108.
- *3.7.4 TCU_STATUS register* on page 3-109.
- *3.7.5 TCU_NODE_CTRLn register* on page 3-110.
- *3.7.6 TCU_NODE_STATUSn register* on page 3-111.
- *3.7.7 TCU_SCR register* on page 3-112.
- *3.7.8 TCU_WC_SxLy_CMAX registers* on page 3-113.

3.7.1 TCU_CTRL register

The TCU Control register disables TCU features. If the hit rate of the individual walk cache is too low, you can disable individual walk caches to improve performance in some systems. Do not modify the AUX bits unless directed to do so by Arm.

Configurations

This register is available in all configurations.

Attributes

The TCU_CTRL register attributes are as follows:

Width	32-bit
Functional group	TCU microarchitectural features. See *3.7 TCU microarchitectural registers* on page 3-106.
Address offset	`0x08E00`
Type	RW
Reset value	`0`

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

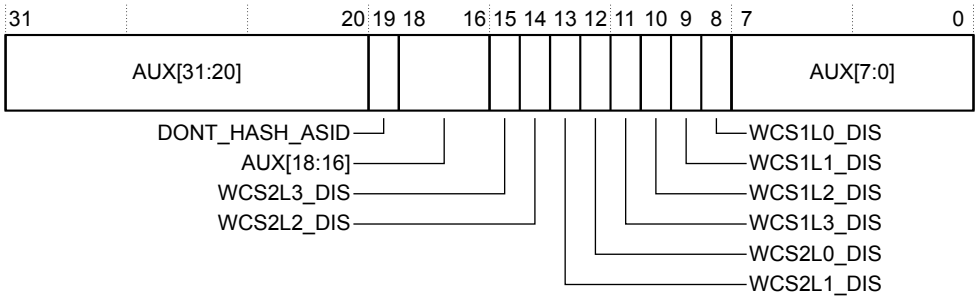

Figure 3-1 TCU_CTRL register bit assignments

The following table shows the bit descriptions.

Table 3-21 TCU_CTRL register bit descriptions

Bits	Name	Description
[31:20]	AUX[31:20]	Reads the value that is written, but has no other effect
[19]	DONT_HASH_ASID	When set to 1, ASID is not used in the hash to create walk cache indices
[18:16]	AUX[18:16]	Reads the value that is written, but has no other effect
[15]	WCS2L3_DIS	Walk cache disable. When a bit of this field is set, it disables the corresponding stage and level of walk cache. WCS2L3_DIS is in bit [15], through to WCS1L0_DIS that is in bit [8].
[14]	WCS2L2_DIS	
[13]	WCS2L1_DIS	
[12]	WCS2L0_DIS	
[11]	WCS1L3_DIS	
[10]	WCS1L2_DIS	
[9]	WCS1L1_DIS	
[8]	WCS1L0_DIS	
[7:0]	AUX[7:0]	Reads the value written, but has no other effect

3.7.2 TCU_QOS register

This register selects the QoS value to attach to transactions issued from the TCU.

Note

The QoS value that is associated with each transaction does not take account of other transactions that are blocked behind it. For example, although translations with a high priority setting in TCU_NODE_CTRL*n* are normally progressed before translations with a lower priority, it is possible for a low-priority page table walk to block a higher priority page table walk from being issued from the TCU.

Configurations

This register is available in all configurations.

Attributes

The TCU_QOS register attributes are as follows:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Width 32-bit

Functional group TCU microarchitectural features. See *3.7 TCU microarchitectural registers on page 3-106*.

Address offset `0x08E04`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

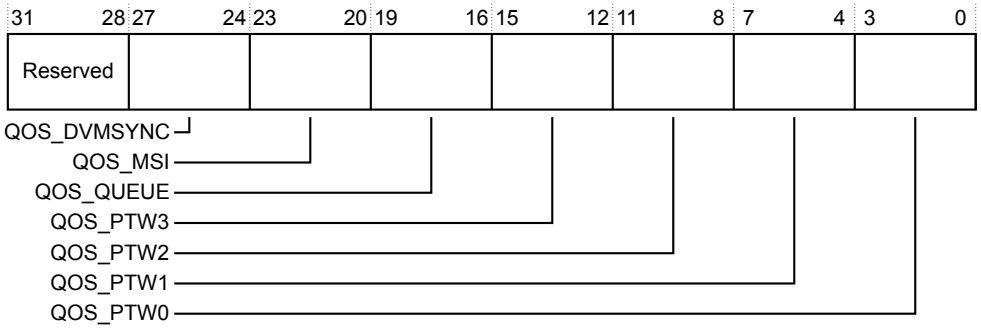

Figure 3-2 TCU_QOS register bit assignments

The following table shows the bit descriptions.

Table 3-22 TCU_QOS register bit descriptions

Bits	Name	Description
[31:28]	-	Reserved
[27:24]	QOS_DVMSYNC	QoS level to use for DVM Sync Completion messages
[23:20]	QOS_MSI	QoS level to use for MSIs
[19:16]	QOS_QUEUE	QoS level to use for queue accesses
[15:12]	QOS_PTW3	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 3
[11:8]	QOS_PTW2	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 2
[7:4]	QOS_PTW1	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 1
[3:0]	QOS_PTW0	QoS level to use for translation table walks for translations that are requested from nodes with TCU_NODE_CTRL*n*.PRIORITY = 0

3.7.3 TCU_CFG register

This section describes the TCU Configuration Information register.

Configurations

This register is available in all configurations.

Attributes

The TCU_CFG register attributes are as follows:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Width 32-bit

Functional group TCU microarchitectural features. See *3.7 TCU microarchitectural registers* on page 3-106.

Address offset `0x08E08`

Type RO

Bit descriptions

The following figure shows the bit assignments.

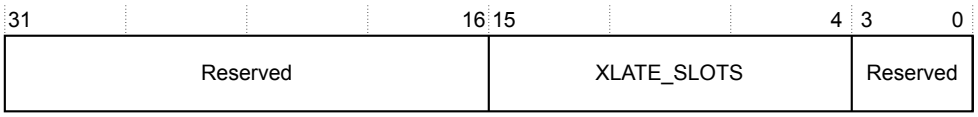

Figure 3-3 TCU_CFG register bit assignments

The following table shows the bit descriptions.

Table 3-23 TCU_CFG register bit descriptions

Bits	Name	Description
[31:16]	-	Reserved
[15:4]	XLATE_SLOTS	Number of translation slots that are available to be shared between all nodes. The value is `TCUCFG_XLATE_SLOTS`. See *2.5.2 TCU buffer configuration options* on page 2-75.
[3:0]	-	Reserved

3.7.4 TCU_STATUS register

This section describes the TCU Status Information register.

Configurations

This register is available in all configurations.

Attributes

The TCU_STATUS register attributes are as follows:

Width 32-bit

Functional group TCU microarchitectural features. See *3.7 TCU microarchitectural registers* on page 3-106.

Address offset `0x08E10`

Type RO

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

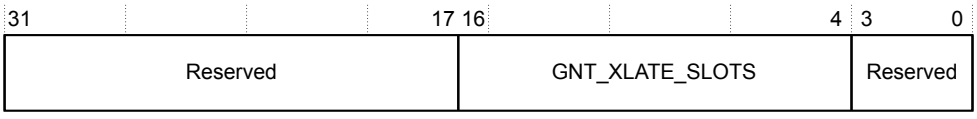

Figure 3-4 TCU_STATUS register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-24 TCU_STATUS register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16:4]	GNT_XLATE_SLOTS	Number of translation slots that are currently allocated to connected nodes. This information can be useful for debugging purposes.
[3:0]	-	Reserved

3.7.5 TCU_NODE_CTRLn register

The TCU_NODE_CTRLn register controls how the TCU communicates with a single DTI master, either a TBU or a PCIe Root Complex implementing ATS.

Each DTI master has a node ID, with the control register for:

- Node 0 at address `0x09000`
- Node 1 at address `0x09004`

The number of registers that are implemented corresponds to the value of `TCUCFG_NUM_TBU`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Configurations

This register is available in all configurations.

Attributes

The TCU_NODE_CTRL*n* register attributes are as follows:

Width 32-bit

Functional group TCU microarchitectural features. See *3.7 TCU microarchitectural registers* on page 3-106.

Address offset `0x09000-0x093FC`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

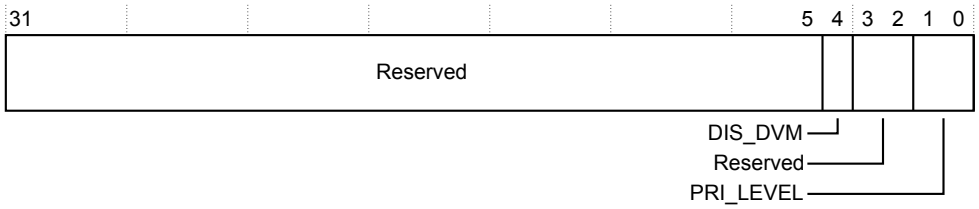

Figure 3-5 TCU_NODE_CTRL register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-25 TCU_NODE_CTRL*n* register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved
[4]	DIS_DVM	Disable DVM. When this bit is set, the node does not participate in DVM invalidation. This setting can improve performance if the node can be slow to respond to invalidations issued over DTI. This bit is only used for TBU nodes. It is ignored for ATS nodes.
[3:2]	-	Reserved
[1:0]	PRI_LEVEL	Priority level. Translation requests from a node with a higher priority level are normally progressed before translation requests from a node with a lower priority level.

3.7.6 TCU_NODE_STATUSn register

The TCU_NODE_STATUSn register provides status for each node, similarly to TCU_NODE_CTRL*n*. Each node has a single status register.

The number of registers that are implemented corresponds to the value of `TCUCFG_NUM_TBU`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Configurations

This register is available in all configurations.

Attributes

The TCU_NODE_STATUS*n* register attributes are as follows:

Width 32-bit

Functional group TCU microarchitectural features. See *3.7 TCU microarchitectural registers* on page 3-106.

Address offset `0x09400-0x097FC`

Type RO

Bit descriptions

The following figure shows the bit assignments.

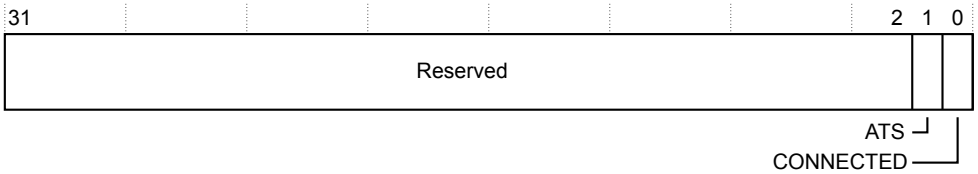

Figure 3-6 TCU_NODE_STATUS register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-26 TCU_NODE_STATUS*n* register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	ATS	Indicates whether the node implements ATS: `0` The node is a TBU connected using DTI-TBU `1` The node is a PCIe Root Complex supporting ATS, connected using DTI-ATS This bit is only valid when CONNECTED = 1. When CONNECTED = 0, this bit is 0.
[0]	CONNECTED	Indicates whether the DTI link for this node is in the connected state: `0` Node currently not in the connected state, including the states transitioning to and from connected state `1` Node currently in the connected state When not connected, write accesses to TBU registers are ignored and read accesses return 0. However, the state might change between reading this register and attempting to access the TBU.

3.7.7 TCU_SCR register

The TCU Secure Control register controls whether Non-secure software is permitted to access each TCU register group.

This register does not control Secure access to the Performance Monitor registers. The SMMU_PMCG_SCR register controls Secure access to these registers as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

Configurations

This register is available in all configurations.

Attributes

The TCU_SCR register attributes are as follows:

Width 32-bit

Functional group TCU microarchitectural features. See *3.7 TCU microarchitectural registers* on page 3-106.

Address offset `0x08E18`

Type Secure, RW

Reset value **sec_override**. See *A.1.12 TCU tie-off signals* on page Appx-A-179.

Bit descriptions

The following figure shows the bit assignments.

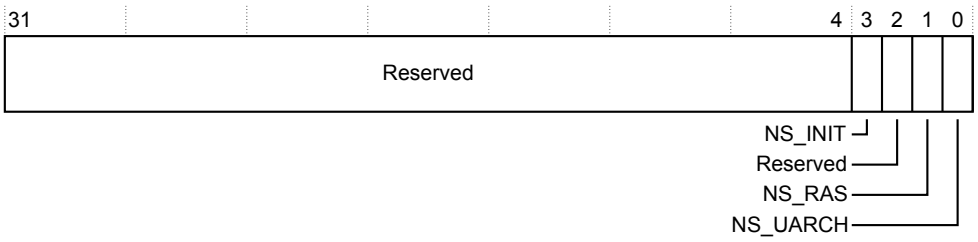

Figure 3-7 TCU_SCR register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-27 TCU_SCR register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	NS_INIT	Non-secure register access that is permitted to the SMMU_S_INIT register
[2]	-	Reserved
[1]	NS_RAS	Non-secure register access that is permitted for RAS registers. When this bit is 0, Non-secure writes to the following register addresses are ignored, and Non-secure reads return zero: `0x08E80-0x08EC0`. The **sec_override** input sets the reset value of this signal. See *A.1.12 TCU tie-off signals* on page Appx-A-179.

[0]	NS_UARCH	Non-secure register access is permitted for microarchitectural registers
		When this bit is 0, Non-secure writes to the following register addresses are ignored, and Non-secure reads return zero:
		`0x08E00-0x08E7C`
		`0x09000-0x093FC`
		The **sec_override** input sets the reset value of this signal. See *A.1.12 TCU tie-off signals* on page Appx-A-179.
		If Secure translation might be used, Arm recommends that software does not set this bit.

3.7.8 TCU_WC_SxLy_CMAX registers

TCU_WC_SxLy_CMAX registers enable you to set maximum capacities for the TCU walk cache RAMS, per stage and level.

The encoding of the TCU_WC_SxLy_CMAX registers is the same as the encoding for the MPAMCFG_CMAX registers that the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A* defines. These registers are readable and writeable registers.

The following table describes the TCU_WC_SxLy_CMAX registers.

Table 3-28 TCU_WC_SxLy_CMAX registers

Address	Name	Field	Position	Meaning
`0x09800`	TCU_WC_S1L0_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 1 level 0
`0x09804`	TCU_WC_S1L1_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 1 level 1
`0x09808`	TCU_WC_S1L2_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 1 level 2
`0x0980C`	TCU_WC_S1L3_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 1 level 3
`0x09810`	TCU_WC_S2L0_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 2 level 0
`0x09814`	TCU_WC_S2L1_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 2 level 1
`0x09818`	TCU_WC_S2L2_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 2 level 2
`0x981C`	TCU_WC_S2L3_CMAX	CMAX	[7:0]	Maximum capacity for TCU Walk Cache stage 2 level 3

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.8 TCU RAS registers

This section describes *Reliability, Availability, and Serviceability* (RAS).

The RAS registers implement the RAS Extension registers, single record format.

Non-secure accesses to these registers, when `TCU_SCR.NS_RAS` = 0, are RAZ/WI. See *3.7.7 TCU_SCR register* on page 3-112.

The RAS registers enable software to monitor the following classes of error:

- *Corrected Errors* (CEs) in the RAMs used by the configuration cache.
- CEs in the RAMs used by the walk caches.

This section contains the following subsections:

3.8.1 TCU_ERRFR register

Use the TCU Error Feature register to discover how the TCU handles errors.

Configurations

This register is available in all configurations.

Attributes

The TCU_ERRFR register attributes are as follows:

Width 32-bit

Functional group *Reliability, Availability, and Serviceability* (RAS). See *3.8 TCU RAS registers* on page 3-114.

Address offset `0x08E80`

Type S, RO

Bit descriptions

The following figure shows the bit assignments.

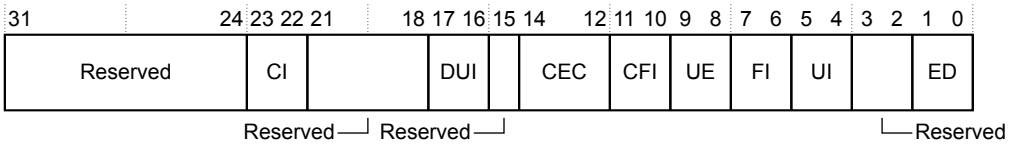

Figure 3-8 TCU_ERRFR register bit assignments

The following table shows the bit descriptions.

Table 3-29 TCU_ERRFR register bit descriptions

Bits	Name	Description	Value
[31:24]	-	Reserved	-
[23:22]	CI	Critical Error Interrupt is always enabled	`01b`
[21:18]	-	Reserved	-

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-29 TCU_ERRFR register bit descriptions (continued)

Bits	Name	Description	Value
[17:16]	DUI	Does not support this feature	00b
[15]	-	Reserved	-
[14:12]	CEC	Does not implement the standard corrected error counter model	000b
[11:10]	CFI	Does not support this feature	00b
[9:8]	UE	In-band error signaling feature is always enabled	01b
[7:6]	FI	Fault handling interrupt is controllable	10b
[5:4]	UI	Error Recovery Interrupt always enabled for UE	01b
[3:2]	-	Reserved	-
[1:0]	ED	Error detection is always enabled	01b

3.8.2 TCU_ERRCTLR register

Use the TCU Error Control register to enable fault handling interrupts.

Configurations

This register is available in all configurations.

Attributes

The TCU_ERRCTLR register attributes are as follows:

Width 32-bit

Functional group *Reliability, Availability, and Serviceability* (RAS). See *3.8 TCU RAS registers* on page 3-114.

Address offset 0x08E88

Type S, RW

Reset value 1

Bit descriptions

The following figure shows the bit assignments.

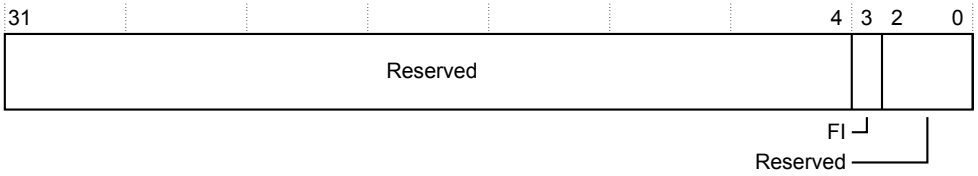

Figure 3-9 TCU_ERRCTLR register bit assignments

The following table shows the bit descriptions.

Table 3-30 TCU_ERRCTLR register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	FI	Fault handling interrupt enable
[2:0]	-	Reserved

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.8.3 TCU_ERRSTATUS register

Use the TCU Error Control register to enable fault handling interrupts.

Certain bits in this register are cleared by writing a 1 to their bit position. These writes are ignored in certain circumstances to avoid race conditions where a new error has occurred which software has not yet observed.

Configurations

This register is available in all configurations.

Attributes

The TCU_ERRSTATUS register attributes are as follows:

Width 32-bit

Functional group *Reliability, Availability, and Serviceability* (RAS). See *3.8 TCU RAS registers* on page 3-114.

Address offset `0x08E90`

Type Secure, RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

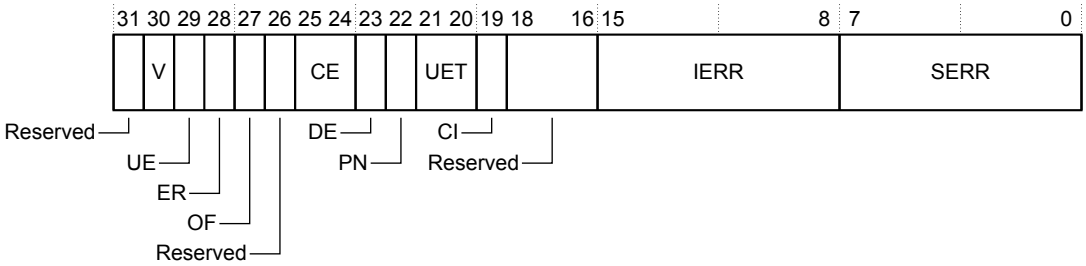

Figure 3-10 TCU_ERRSTATUS register bit assignments

The following table shows the bit descriptions.

Table 3-31 TCU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are: `0` ERRSTATUS is not valid `1` ERRSTATUS is valid. At least one error has been recorded If any of the UE, DE, or CE bits are set to 1, and are not being cleared to 0 in the same write, direct writes to this bit are ignored. This bit is read/write-one-to-clear. This bit resets to zero on a reset.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-31 TCU_ERRSTATUS register bit descriptions (continued)

Bits	Name	Description
[29]	UE	Uncorrected error, or errors. The possible values of this bit are: `0` No errors that could not be corrected or deferred `1` At least one error detected that has not been corrected or deferred Direct writes to this bit are ignored if the OF bit is set to 1 and is not being cleared to zero in the same write. This bit is read/write-one-to-clear.
[28]	ER	Error Reported. The possible values of this bit are: `0` No in-band error (External abort) is reported `1` The node to the master making the access or other transaction signaled an External abort Writes to this field are ignored.
[27]	OF	Overflow. Multiple errors are detected. This bit is set to 1 when: • An Uncorrected error is detected and the previous error syndrome is kept because UE == 1 • A Deferred error is detected and the previous error syndrome is discarded because DE == 1 • A Corrected error is detected and the CE field might be updated for the new Corrected error • A Deferred error is detected and UE == 1 • A Corrected error is detected and either or both the DE or UE bits are set to 1 This bit is cleared by writing a 1 to it. A write of 0 is ignored.
[26]	-	Reserved
[25:24]	CE	Correctable Error, or errors. `00b` No correctable errors recorded `10b` At least one Corrected error recorded Other values are Reserved. This field is cleared by writing `11b` to it. If OF is set and not being cleared, the write is ignored. A write of any value other than `11b` is ignored.
[23]	DE	Deferred error, or errors. The possible values of this bit are: `0` No errors were deferred `1` At least one error was not corrected and deferred This error is raised when **wpoison** is set in BIU. If the OF bit is set to 1 and is not being cleared to 0 in the same write, direct writes to this bit are ignored. This bit is read/write-one-to-clear.
[22]	PN	Poison. The possible values of this bit are: `0` Uncorrected error or deferred error is recorded because a corrupt value was detected, for example, by an *Error Detection Code* (EDC) `1` Uncorrected error or deferred error is recorded because a poison value was detected Writes to this field are ignored.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-31 TCU_ERRSTATUS register bit descriptions (continued)

Bits	Name	Description
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. The possible values of this field are: 0b00 Uncorrected error, Uncontainable error (UC) 0b11 Uncorrected error, Signaled or Recoverable error (UER)
[19]	CI	Indicates whether a critical error condition has been recorded. The possible values of this bit are: 0 No critical error condition 1 Critical error condition Writes to this field are ignored.
[18:16]	-	Reserved
[15:8]	IERR	IMPLEMENTATION DEFINED error code. When SERR≠0, this field indicates the source of the error: 12h PIU CMD RPOISON 11h TMU CCB MCC DATA 10h TMU CCB MCC TAGS 0Fh TMU WCB MWC DATA 0Eh TMU WCB MWC TAGS 0Dh TMU CCB MCC REPL 0Ch TMU CCB MCC PCNT 0Bh TMU CCB MCC PLIM 0Ah TMU WCB MWC REPL 09h TMU WCB MWC PCNT 08h TMU WCB MWC PLIM 07h Reserved 06h Reserved 05h TMU HTTU RAM 04h TMU TWB WMB SCRATCH 03h TMU TWB WMB WLK STATUS 02h TMU TWB WMB LKP STATUS 01h TMU HZU PTR 00h TMU TWB BSU Writes to this field are ignored.
[7:0]	SERR	The error code provides information about the earliest unacknowledged error. It can contain the following values: 2 Single or double error from RAMs that are not CCB or WCB TAGS or DATA 8 Single or double error from CCB or WCB data 9 Single or double error from CCB or WCB tags 21 Poisoned data read from downstream All other values are reserved. Writes to this field are ignored.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.8.4 TCU_ERRGEN register

Use the TCU Error Generation Register to test software for when a RAS error occurs in the RAM.

The field locations are same as for *3.15.3 TBU_ERRSTATUS register* on page 3-149.

When this register is updated, the TCU_ERRSTATUS register is also updated with the same value, as long as the write data generates a valid error record.

A write to ERRGEN is valid if all the following is true:

- ERRGEN.V is set
- At least one of the following is true (CE is legal if CE == `2'b00` or CE == `2'b10`):
  - ERRGEN.UE is set and CE is legal
  - ERRGEN.DE is set and CE is legal
  - ERRGEN.CE is set to `2'b10`
- One of the following is true:
  - UET == `2'b00`
  - UET == `2'b11` and UE == 1
- If a valid error record is written, then the appropriate interrupt, or interrupts, are also generated.

This register has identical fields to TCU_ERRSTATUS. See *3.15.3 TBU_ERRSTATUS register* on page 3-149.

Configurations

This register is available in all configurations.

Attributes

The TCU_ERRGEN register attributes are as follows:

Width	64-bit
Functional group	*Reliability, Availability, and Serviceability* (RAS). See *3.8 TCU RAS registers* on page 3-114.
Address offset	`0x08EC0`
Type	Secure, RW
Reset value	`0`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.9 TCU system discovery registers

This section describes the TCU system discovery registers.

This section contains the following subsections:

3.9.1 TCU_SYSDISC0 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC0 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E34`

Type RO

Reset value `TCUCFG_WC_DEPTH`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

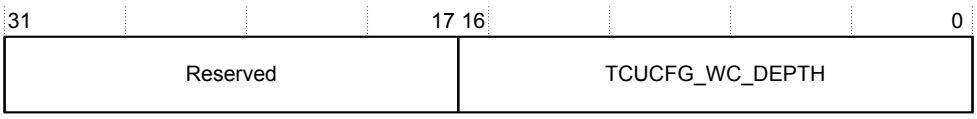

Figure 3-11 TCU_SYSDISC0 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-32 TCU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved

[16:0]	TCUCFG_WC_DEPTH	The read data reflects the chosen parameter value, for example:
		`17'h0_0008 : 8`
		….
		`17'h1_0000 : 65536`

3.9.2 TCU_SYSDISC1 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC1 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E38`

Type RO

Reset value `TCUCFG_CC_DEPTH`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

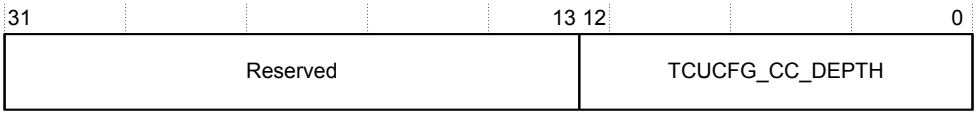

Figure 3-12 TCU_SYSDISC1 register bit assignments

The following table shows the bit descriptions.

Table 3-33 TCU_SYSDISC1 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved

[12:0]	TCUCFG_CC_DEPTH	The read data reflects the chosen parameter value, for example:
		`13'h0004 : 4`
		….
		`13'h1000 : 4096`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.9.3 TCU_SYSDISC2 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC2 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E3C`

Type RO

Reset value `TCUCFG_WC_WAYS`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

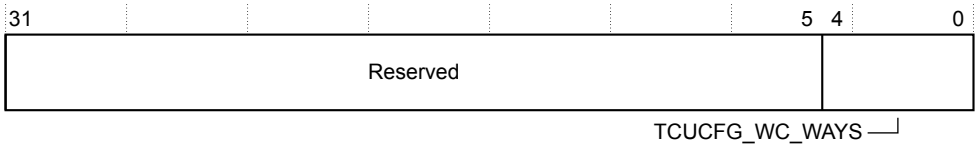

Figure 3-13 TCU_SYSDISC2 register bit assignments

The following table shows the bit descriptions.

Table 3-34 TCU_SYSDISC2 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved

[4:0]	TCUCFG_WC_WAYS	The read data reflects the chosen parameter value, for example:
		`5'h04 : 4`
		….
		`5'h10 : 16`

3.9.4 TCU_SYSDISC3 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC3 register attributes are as follows:

Width 32-bit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E40`

Type RO

Reset value `TCUCFG_WC_BANKS`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

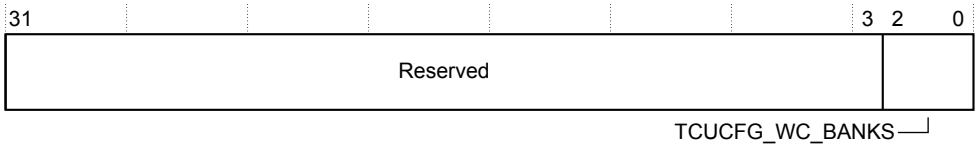

Figure 3-14 TCU_SYSDISC3 register bit assignments

The following table shows the bit descriptions.

Table 3-35 TCU_SYSDISC3 register bit descriptions

Bits	Name	Description
[31:3]	-	Reserved

[2:0]	TCUCFG_WC_BANKS	The read data reflects the chosen parameter value, for example:
		`3'b001 : 1`
		….
		`3'b100 : 4`

3.9.5 TCU_SYSDISC4 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC4 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E44`

Type RO

Reset value `TCUCFG_XLATE_SLOTS`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

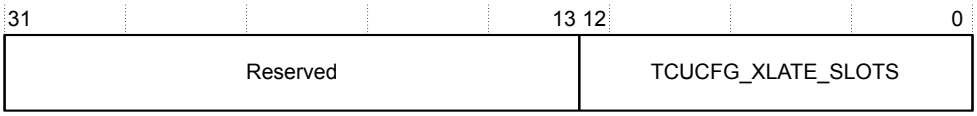

Figure 3-15 TCU_SYSDISC4 register bit assignments

The following table shows the bit descriptions.

Table 3-36 TCU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved

[12:0]	TCUCFG_XLATE_SLOTS	The read data reflects the chosen parameter value, for example:
		`13'h0004 : 4`
		….
		`13'h1000 : 4096`

3.9.6 TCU_SYSDISC5 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC5 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E48`

Type RO

Reset value `TCUCFG_PTW_SLOTS`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

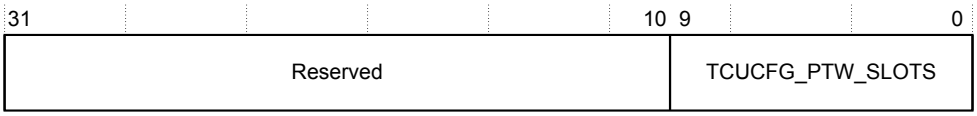

Figure 3-16 TCU_SYSDISC5 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-37 TCU_SYSDISC5 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved

[9:0]	TCUCFG_PTW_SLOTS	The read data reflects the chosen parameter value, for example:
		`9'h002 : 2`
		….
		`9'h200 : 512`

3.9.7 TCU_SYSDISC6 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC6 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E4C`

Type RO

Reset value `TCUCFG_CTW_SLOTS`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

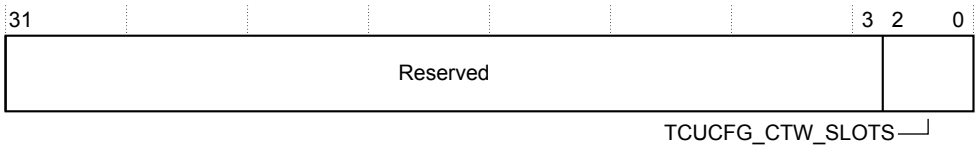

Figure 3-17 TCU_SYSDISC6 register bit assignments

The following table shows the bit descriptions.

Table 3-38 TCU_SYSDISC6 register bit descriptions

Bits	Name	Description
[31:3]	-	Reserved

[2:0]	TCUCFG_CTW_SLOTS	The read data reflects the chosen parameter value, for example:
		`3'b001 : 1`
		….
		`3'b100 : 4`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.9.8 TCU_SYSDISC7 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC7 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E50`

Type RO

Reset value `TCUCFG_CC_IDXGEN_MODE`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

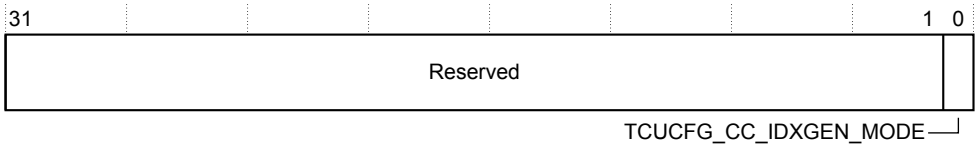

Figure 3-18 TCU_SYSDISC7 register bit assignments

The following table shows the bit descriptions.

Table 3-39 TCU_SYSDISC7 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved

[0]	TCUCFG_CC_IDXGEN_MODE	The read data reflects the chosen parameter value, for example:
		`1'b0 : 0`
		….
		`1'b1 : 1`

3.9.9 TCU_SYSDISC8 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC8 register attributes are as follows:

Width 32-bit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E54`

Type RO

Reset value `TCUCFG_DTI_ATS`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

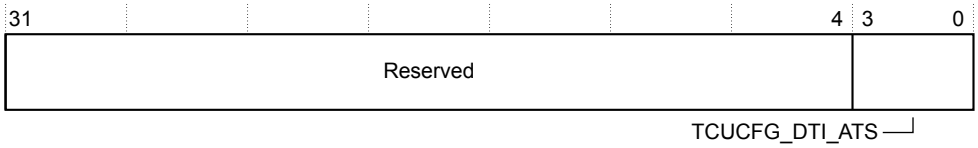

Figure 3-19 TCU_SYSDISC8 register bit assignments

The following table shows the bit descriptions.

Table 3-40 TCU_SYSDISC8 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved

[3:0]	TCUCFG_DTI_ATS	The read data reflects the chosen parameter value, for example:
		`4'b0000 : 0`
		….
		`4'b1000 : 8`

3.9.10 TCU_SYSDISC9 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC9 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E58`

Type RO

Reset value `TCU_CFG_NUM_TBU`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

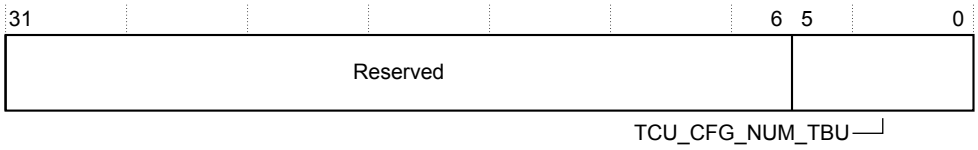

Figure 3-20 TCU_SYSDISC9 register bit assignments

The following table shows the bit descriptions.

Table 3-41 TCU_SYSDISC9 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved

[5:0]	TCU_CFG_NUM_TBU	The read data reflects the chosen parameter value, for example:
		`6'h01 : 1`
		….
		`6'h3E : 62`

3.9.11 TCU_SYSDISC10 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC10 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers on page 3-120*.

Address offset `0x08E5C`

Type RO

Reset value `TCUCFG_NUM_PMU_COUNTERS`. See *2.5.2 TCU buffer configuration options on page 2-75*.

Bit descriptions

The following figure shows the bit assignments.

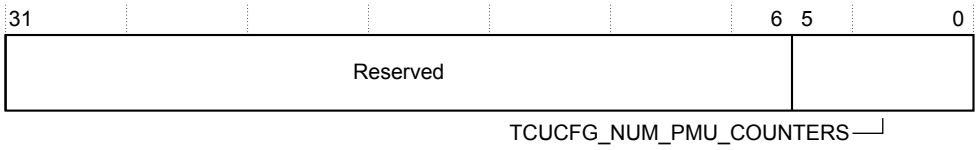

Figure 3-21 TCU_SYSDISC10 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-42 TCU_SYSDISC10 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved

[5:0]	TCUCFG_NUM_PMU_COUNTERS	The read data reflects the chosen parameter value, for example:
		`6'h04 : 4`
		….
		`6'h20 : 32`

3.9.12 TCU_SYSDISC11 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC11 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E60`

Type RO

Reset value `TCUCFG_PARTID_WIDTH`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

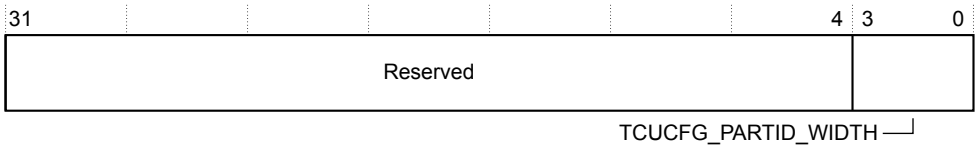

Figure 3-22 TCU_SYSDISC11 register bit assignments

The following table shows the bit descriptions.

Table 3-43 TCU_SYSDISC11 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved

[3:0]	TCUCFG_PARTID_WIDTH	The read data reflects the chosen parameter value, for example:
		`4'b0001 : 1`
		….
		`4'b1001 : 9`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.9.13 TCU_SYSDISC12 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC12 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E64`

Type RO

Reset value `TCUCFG_HZU_DEPTH`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

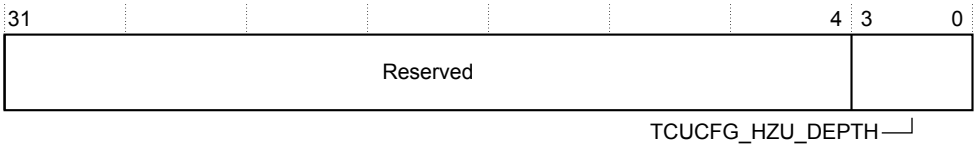

Figure 3-23 TCU_SYSDISC12 register bit assignments

The following table shows the bit descriptions.

Table 3-44 TCU_SYSDISC12 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved

[3:0]	TCUCFG_HZU_DEPTH	The read data reflects the chosen parameter value, for example:
		`7'h02 : 2`
		….
		`7'h40 : 64`

3.9.14 TCU_SYSDISC13 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC13 register attributes are as follows:

Width 32-bit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E68`

Type RO

Reset value `TCUCFG_PREFETCH_SUPPORTED`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

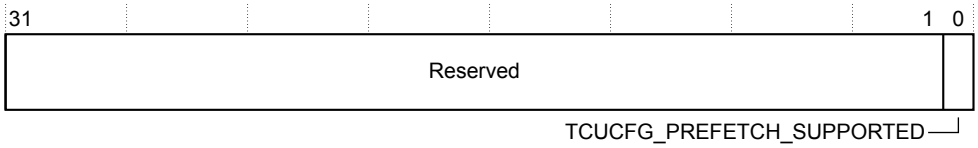

Figure 3-24 TCU_SYSDISC13 register bit assignments

The following table shows the bit descriptions.

Table 3-45 TCU_SYSDISC13 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved

[0]	TCUCFG_PREFETCH_SUPPORTED	The read data reflects the chosen parameter value, for example:
		`1'b0 : 0`
		….
		`1'b1 : 1`

3.9.15 TCU_SYSDISC14 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC14 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E6C`

Type RO

Reset value `TCUCFG_DATARAM_TYPE`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

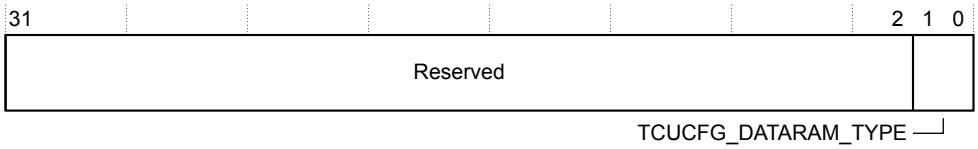

Figure 3-25 TCU_SYSDISC14 register bit assignments

The following table shows the bit descriptions.

Table 3-46 TCU_SYSDISC14 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TCUCFG_DATARAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2'b00 : 0`
		….
		`2'b10 : 2`

3.9.16 TCU_SYSDISC15 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC15 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E70`

Type RO

Reset value `TCUCFG_SLOTRAM_TYPE`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

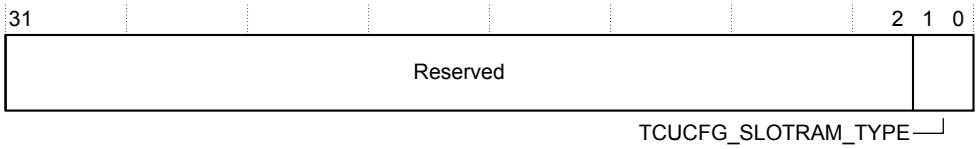

Figure 3-26 TCU_SYSDISC15 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-47 TCU_SYSDISC15 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TCUCFG_SLOTRAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2'b00 : 0`
		….
		`2'b10 : 2`

3.9.17 TCU_SYSDISC16 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC16 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E74`

Type RO

Reset value `TCUCFG_CACHERAM_TYPE`. See *2.5.2 TCU buffer configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

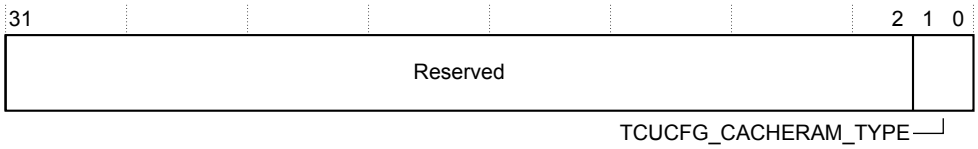

Figure 3-27 TCU_SYSDISC16 register bit assignments

The following table shows the bit descriptions.

Table 3-48 TCU_SYSDISC16 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TCUCFG_CACHERAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2'b00 : 0`
		….
		`2'b01 : 1`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.9.18 TCU_SYSDISC17 system discovery register

The TCU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TCU_SYSDISC17 register attributes are as follows:

Width 32-bit

Functional group TCU system discovery registers. See *3.9 TCU system discovery registers* on page 3-120.

Address offset `0x08E78`

Type RO

Reset value `TCUCFG_QTW_DATA_WIDTH`. See *2.5.1 TCU I/O configuration options* on page 2-75.

Bit descriptions

The following figure shows the bit assignments.

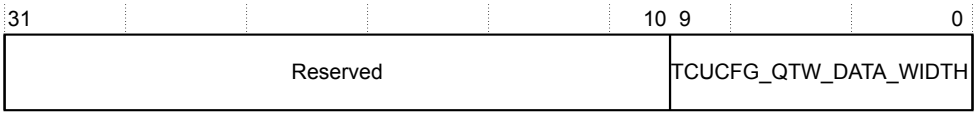

Figure 3-28 TCU_SYSDISC17 register bit assignments

The following table shows the bit descriptions.

Table 3-49 TCU_SYSDISC17 register bit descriptions

Bits	Name	Description
[31:10]	-	Reserved

[9:0]	TCUCFG_QTW_DATA_WIDTH	The read data reflects the chosen parameter value, for example:
		`10'h040 : 64`
		….
		`10'h200 : 512`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.10 TCU PIU integration registers

This section describes the *Programmer Interface Unit* (PIU) integration registers.

This section contains the following subsections:

3.10.1 ITEN register for the TCU

Integration mode register for the TCU.

Configurations

This register is available in all configurations.

Attributes

The ITEN register attributes are as follows:

Width 32-bit

Functional group Performance monitor. See *3.8 TCU RAS registers* on page 3-114.

Address offset `0x08E20`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

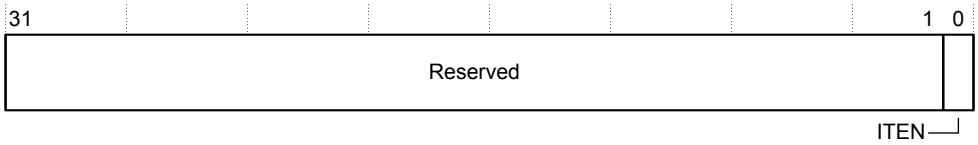

Figure 3-29 ITEN register bit assignments

The following table shows the bit descriptions.

Table 3-50 ITEN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	ITEN	`0` Integration mode is disabled `1` Integration mode is enabled

3.10.2 ITOP register for the TCU Programmer Interface Unit

This section describes the TCU ITOP register for the *Programmer Interface Unit* (PIU).

Configurations

This register is available in all configurations.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Attributes

The ITOP register attributes are as follows:

Width 32-bit

Functional group *Programmer Interface Unit* (PIU) integration registers. See *3.10 TCU PIU integration registers* on page 3-135.

Address offset `0x08E24`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

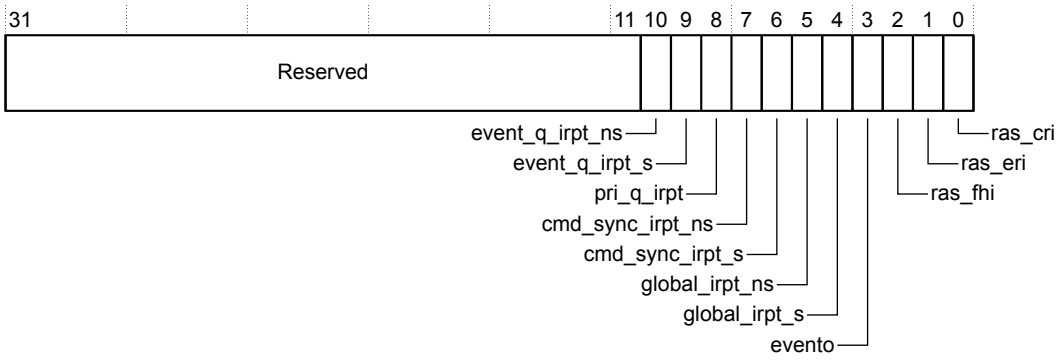

Figure 3-30 ITOP register bit assignments

The following table shows the bit descriptions.

Table 3-51 ITOP register bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, SBZ
[10]	event_q_irpt_ns	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **event_q_irpt_ns**
[9]	event_q_irpt_s	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **event_q_irpt_s**
[8]	pri_q_irpt	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **pri_q_irpt**
[7]	cmd_sync_irpt_ns	`0` When ITEN.ITEN == `0`, SBZ. `1` When ITEN.ITEN == `1`, the value driven to **cmd_sync_irpt_ns**
[6]	cmd_sync_irpt_s	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **cmd_sync_irpt_s**
[5]	global_irpt_ns	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **global_irpt_ns**

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-51 ITOP register bit descriptions (continued)

Bits	Name	Description
[4]	global_irpt_s	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **global_irpt_s**
[3]	evento	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **evento**
[2]	ras_fhi	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **ras_fhi**
[1]	ras_eri	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **ras_eri**
[0]	ras_cri	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **ras_cri**

See:

- *A.1.9 TCU interrupt signals* on page Appx-A-175
- *A.1.11 TCU event interface signal* on page Appx-A-177

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.11 TCU TMU integration registers

This section describes the TCU *Translation Management Unit* (TMU) integration registers.

This section contains the following subsections:

3.11.1 ITOP register for the TCU Translation Management Unit

This section describes the ITOP register for the TCU *Translation Management Unit* (TMU).

Configurations

This register is available in all configurations.

Attributes

The ITOP register attributes are as follows:

Width 32-bit

Functional group TCU *Translation Management Unit* (TMU) integration registers. See *3.11 TCU TMU integration registers* on page 3-138.

Address offset `0x08E2C`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

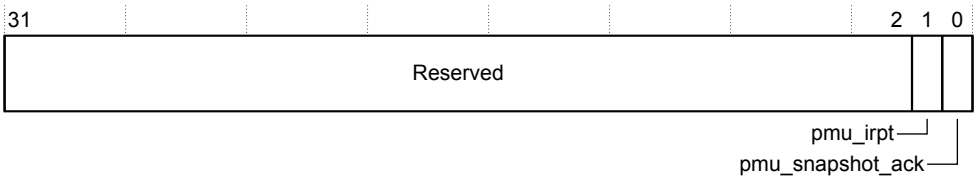

Figure 3-31 ITOP register bit assignments

The following table shows the bit descriptions.

Table 3-52 ITOP register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved, SBZ
[1]	pmu_irpt	`0` When ITEN.ITEN == 0, SBZ `1` When ITEN.ITEN == 1, the value driven to **pmu_irpt**
[0]	pmu_snapshot_ack	`0` When ITEN.ITEN == 0, SBZ `1` When ITEN.ITEN == 1, the value driven to **pmusnapshot_ack**

See:

- *A.1.9 TCU interrupt signals* on page Appx-A-175
- *A.1.5 TCU PMU snapshot interface signals* on page Appx-A-174

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.11.2 ITIN register for the TCU Translation Management Unit

This section describes the ITIN register for the TCU *Translation Management Unit* (TMU).

Configurations

This register is available in all configurations.

Attributes

The ITIN register attributes are as follows:

Width 32-bit

Functional group TCU *Translation Management Unit* (TMU) integration registers. See *3.11 TCU TMU integration registers* on page 3-138.

Address offset `0x08E30`

Type RO

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

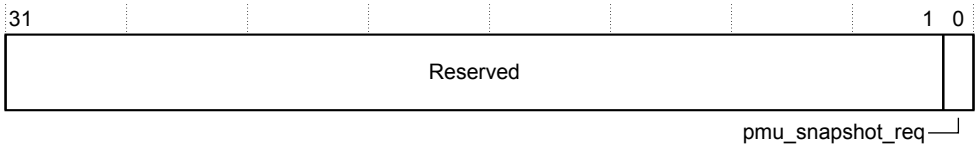

Figure 3-32 ITIN register bit assignments

The following table shows the bit descriptions.

Table 3-53 ITIN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, SBZ
[0]	pmu_snapshot_req	`0` When ITEN.ITEN == 0, SBZ `1` When ITEN.ITEN == 1, reflects **pmusnapshot_req**

See *A.1.5 TCU PMU snapshot interface signals* on page Appx-A-174.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.12 TBU component and peripheral ID registers

This section describes the TBU component and peripheral ID registers.

The following table shows the TBU component and peripheral ID.

Table 3-54 TBU component and peripheral ID registers

Name	Offset	Field	Value	Description
SMMU_CIDR3, Component ID3	0x00FFC	[7:0]	0xB1	Preamble
SMMU_CIDR2, Component ID2	0x00FF8	[7:0]	0x05	Preamble
SMMU_CIDR1, Component ID1	0x00FF4	[7:0]	0xF0	Preamble
SMMU_CIDR0, Component ID0	0x00FF0	[7:0]	0x0D	Preamble
SMMU_PIDR3, Peripheral ID3	0x00FEC	[7:4]	MAX(*p_level*, **ecorevnum**)	REVAND, minor revision. Where *p_level* is: **0** For p0 **1** For p1
		[3:0]	0x00	CMOD
SMMU_PIDR2, Peripheral ID2	0x00FE8	[7:4]	0x00	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
SMMU_PIDR1, Peripheral ID1	0x00FE4	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PIDR0, Peripheral ID0	0x00FE0	[7:0]	0x88	PART_0: bits [7:0] of the Part number
SMMU_PIDR7, Peripheral ID7	0x00FDC	-	RES0	Reserved
SMMU_PIDR6, Peripheral ID6	0x00FD8			
SMMU_PIDR5, Peripheral ID5	0x00FD4			
SMMU_PIDR4, Peripheral ID4	0x00FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.13 TBU PMU registers

This section describes the *Performance Monitor Unit* (PMU).

The TBU PMU registers follow the register layout that the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* Performance Monitor Extension describes.

This section contains the following subsections:

3.13.1 Registers

The TBU and TCU support the same PMCG registers.

See *3.6 TCU PMU registers* on page 3-103.

SMMU_PMCG_SMR0 is 24 bits, because the TBU uses 24-bit StreamIDs architecturally, even though a static tie-off sets either 8 bits or 16 bits.

3.13.2 Events

Each event indicates whether the SMMU_PMCG_SMR0 register can filter it. For events that cannot be filtered, whether they are visible only when Secure events are visible by SMMU_PMCG_SCR.SO = 1.

The following table shows the architectural events that are implemented.

Table 3-55 Architectural events

Event ID	Description	Filterable	Secure only	Description
0	Clock cycle	No	No	Counts every clock cycle. Does not count cycles where the clock is gated after a clock Q-Channel handshake.
1	Transaction	Yes	-	Counts once per transaction issued downstream into the system
2	TLB miss that an incoming transaction or translation request causes	Yes	-	Counts once per non-speculative TCU translation request
7	PCIe ATS Translated Transaction passed through SMMU	Yes	-	Counts once per ATS transaction that is issued into the system

The following table shows the IMPLEMENTATION DEFINED events that are implemented.

Table 3-56 IMPLEMENTATION DEFINED events

Event ID	Description	Filterable	Secure only	Description
0x80	Main TLB lookup	Yes	-	Counts once per transaction that accesses the Main TLB
0x81	Main TLB miss	Yes	-	Counts once per transaction that accesses the Main TLB and does not hit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-56 IMPLEMENTATION DEFINED events (continued)

Event ID	Description	Filterable	Secure only	Description
0x82	Main TLB read	Yes	-	Counts once per access to the Main TLB RAMs. A transaction might access the Main TLB multiple times to look for different page sizes.
0x83	Micro TLB lookup	Yes	-	Counts once per lookup in the Micro TLB
0x84	Micro TLB miss	Yes	-	Counts once per miss in the Micro TLB
0x85	Slots full	No	Yes	Counts once per cycle when all slots are occupied and not ready to issue downstream
0x86	Out of translation tokens	No	Yes	Counts once per cycle when a translation request cannot be issued because all translation tokens are in use
0x87	Write data buffer full	No	Yes	Counts once per cycle when a transaction is blocked because the write data buffer is full
0x8B	DCMO downgrade	Yes	-	Counts once whenever either: • A MakeInvalid transaction on TBS is output as CleanInvalid on TBM • A ReadOnceMakeInvalid transaction on TBS is output as ReadOnceCleanInvalid on TBM
0x8C	DCP fail	Yes	-	Counts once whenever either: • An LTI WDCP transaction on the LA channel is downgraded as W on the LR channel. • An LTI DCP transaction on the LA channel is responded to as FaultRAZWI on the LR channel is counted. This response can be because of memory attributes or DCP, R, W, X permission check failure in the TLBU or the DTI fault response with Non-Abort. The transaction responded with FaultAbort because of DTI StreamDisable, GlobalDisable is not counted.

3.13.3 SMMU_PMCG_CFGR

An MMU-700 implementation assumes fixed values for SMMU_PMCG_CFGR, and these values define behavioral aspects of the implementation.

For information about the SMMU_PMCG_CFCR fields values, see *SMMUv3 PMU register architectural options* on page 2-45.

See also *2.5 Configuration options and methodology* on page 2-75.

3.13.4 SMMU_PMCG_CEID{0-1} registers

The SMMU_PMCG_CEID{0-1} registers indicate the architectural events that are supported. They are described as 64-bit registers, but they are accessed 32 bits at a time through the 32-bit DTI register access messages.

The following table shows the SMMU_PMCG_CEID{0-1} registers.

Table 3-57 SMMU_PMCG_CEID{0-1} registers

Name	Offset	Value
SMMU_PMCG_CEID0	0x02e20	0x00000087
SMMU_PMCG_CEID1	0x02e28	0x00000000

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.13.5 PMU ID registers

The PMU ID registers are defined as follows. The PMU ID registers appear only in Performance Monitor Page 0. Page 1 does not contain any ID registers.

The following table shows the PMU ID registers.

Table 3-58 PMU ID registers

Name	Offset	Field	Value	Description
SMMU_PMCG_CIDR3, Component ID3	0x02FFC	[7:0]	0xB1	Preamble
SMMU_PMCG_CIDR2, Component ID2	0x02FF8	[7:0]	0x05	Preamble
SMMU_PMCG_CIDR1, Component ID1	0x02FF4	[7:0]	0x90	Preamble
SMMU_PMCG_CIDR0, Component ID0	0x02FF0	[7:0]	0x0D	Preamble
SMMU_PMCG_PIDR3, Peripheral ID3	0x02FEC	[7:4]	MAX(*p_level*, **ecorevnum**)	REVAND, minor revision, where *p_level* is: 0 For p0 1 For p1
		[3:0]	0x00	CMOD
SMMU_PMCG_PIDR2, Peripheral ID2	0x02FE8	[7:4]	0x00	REVISION, major revision
		[3]	1	JEDEC-assigned value for DES always used
		[2:0]	3	DES_1: bits [6:4] bits of the JEP106 Designer code
SMMU_PMCG_PIDR1, Peripheral ID1	0x02FE4	[7:4]	0xB	DES_0: bits [3:0] of the JEP106 Designer code
		[3:0]	0x4	PART_1: bits [11:8] of the Part number
SMMU_PMCG_PIDR0, Peripheral ID0	0x02FE0	[7:0]	0x88	PART_0: bits [7:0] of the Part number
SMMU_PMCG_PIDR7, Peripheral ID7	0x02FDC	-	RES0	Reserved
SMMU_PMCG_PIDR6, Peripheral ID6	0x02FD8			Reserved
SMMU_PMCG_PIDR5, Peripheral ID5	0x02FD4			Reserved
SMMU_PMCG_PIDR4, Peripheral ID4	0x02FD0	[7:4]	0x0	SIZE = 4KB
		[3:0]	0x4	DES_2: JEP106 Designer continuation code
SMMU_PMCG_PMAUTHSTATUS	0x02FB8	[7:0]	0x00	No authentication interface is implemented

The PMAUTHSTATUS, PMDEVARCH, and PMDEVTYPE registers are implemented as the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2* defines.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.14 TBU microarchitectural registers

You can set the microarchitectural registers at boot time to optimize TBU behavior for your system. Arm recommends that you use the default values for most systems.

The *3.14.2 TBU_SCR register* on page 3-145 is Secure-only. Non-secure access to this register is *Read-As-Zero* (RAZ)/*Writes-Ignored* (WI).

Non-secure access to the *3.14.1 TBU_CTRL register* on page 3-144when TBU_SCR.NS_UARCH = 0 is RAZ/WI.

This section contains the following subsections:

- *3.14.1 TBU_CTRL register* on page 3-144.
- *3.14.2 TBU_SCR register* on page 3-145.

3.14.1 TBU_CTRL register

The TBU_CTRL register disables TBU features. Do not modify the bits in this register unless Arm instructs you to do so.

Configurations

This register is available in all configurations.

Attributes

The TBU_CTRL register attributes are as follows:

Width 32-bit

Functional group TBU microarchitectural features. See *3.14 TBU microarchitectural registers* on page 3-144.

Address offset `0x08E00`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

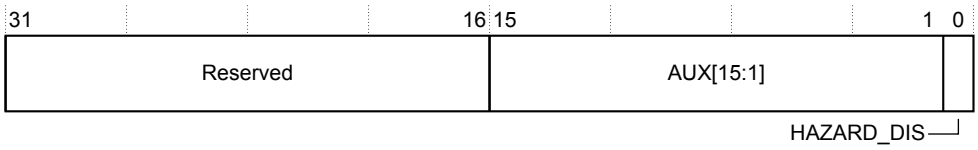

Figure 3-33 TBU_CTRL register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-59 TBU_CTRL register bit descriptions

Bits	Name	Description
[31:20]	-	Reserved
[15:1]	[AUX15:1]	Reads the value that is written, but has no other effect
[0]	HAZARD_DIS	`0` When this bit is clear, and multiple outstanding transactions access the same page, the TBU sends a single translation request and uses that for all the affected transactions. `1` When this bit is set, disables hazarding between translation requests from transactions in the same page. Post reset, this bit can be set to 1 once, but cannot be cleared again without a reset.

3.14.2 TBU_SCR register

This section describes the TBU Secure Control register.

Configurations

This register is available in all configurations.

Attributes

The TBU_SCR register attributes are as follows:

Width 32-bit

Functional group *Reliability, Availability, and Serviceability* (RAS). See *3.8 TCU RAS registers* on page 3-114.

Address offset `0x08EC0`

Type RW

Reset value **sec_override**. See *A.2.9 TBU tie-off signals* on page Appx-A-192.

Bit descriptions

The following figure shows the bit assignments.

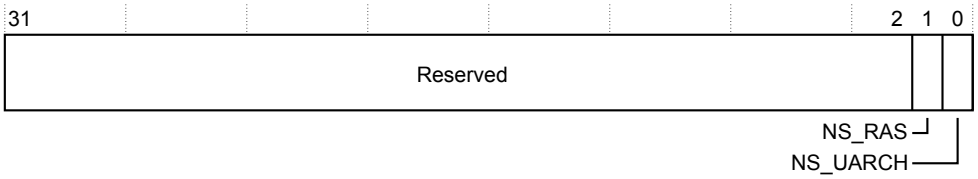

Figure 3-34 TBU_SCR register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-60 TBU_SCR register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved
[1]	NS_RAS	Non-secure register access that is permitted for microarchitectural registers. When this bit is 0, Non-secure writes to the following register addresses are ignored, and Non-secure reads return zero: `0x08E80-0x08EC0` The **sec_override** input sets the reset value of this signal. See *A.2.9 TBU tie-off signals* on page Appx-A-192.
[0]	NS_UARCH	Non-secure register access that is permitted for TBU_CTRL. When this bit is 0, Non-secure writes to TBU_CTRL is ignored, and Non-secure reads return zero. The **sec_override** input sets reset value of this signal. See *A.2.9 TBU tie-off signals* on page Appx-A-192. If Secure translation might be used, Arm recommends that software does not set this bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.15 TBU RAS registers

This section describes *Reliability, Availability, and Serviceability* (RAS) registers.

These registers implement the RAS Extension registers, single record format.

Non-secure accesses to these registers, when TBU_SCR.NS_RAS = 0, are RAZ/WI.

The RAS registers enable software to monitor the following classes of error:

- *Corrected Errors* (CEs) in the RAMs that the Main TLB uses.
- CEs in the RAMs, that the Write Data Buffer uses.

RAS error reporting

When a CE occurs:

A CE is reported in *3.15.3 TBU_ERRSTATUS register* on page 3-149.

If TBU_ERRCTLR.FI is set, an interrupt is raised on **ras_fhi**. See *TBU interrupt interfaces* on page 2-37.

This section contains the following subsections:

3.15.1 TBU_ERRFR register

Error Feature Register.

Configurations

This register is available in all configurations.

Attributes

The TBU_ERRFR register attributes are as follows:

Width 32-bit

Functional group TBU *Reliability, Availability, and Serviceability* (RAS). See *3.15 TBU RAS registers* on page 3-147.

Address offset `0x08E80`

Type Secure, RO

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

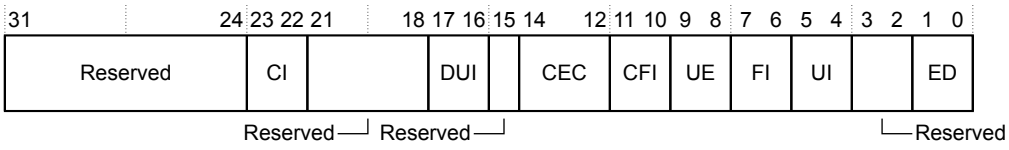

Figure 3-35 TBU_ERRFR register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-61 TBU_ERRFR register bit descriptions

Bits	Name	Description	Value
[31:24]	-	Reserved	-
[23:22]	CI	Critical Error Interrupt is always enabled	01b
[21:18]	-	Reserved	-
[17:16]	DUI	Does not support feature	00b
[15]	-	Reserved	-
[14:12]	CEC	Does not implement the standard corrected error counter model	000b
[11:10]	CFI	Does not support feature	00b
[9:8]	UE	In-band error signaling feature is not enabled	00b
[7:6]	FI	Fault handling interrupt is controllable	10b
[5:4]	UI	Error Recovery Interrupt always enabled for UE	01b
[3:2]	-	Reserved	-
[1:0]	ED	Error detection is always enabled	01b

3.15.2 TBU_ERRCTLR register

Use the TBU Error Control register to enable fault handling interrupts.

Configurations

This register is available in all configurations.

Attributes

The TBU_ERRCTLR register attributes are as follows:

Width 32-bit

Functional group TBU *Reliability, Availability, and Serviceability* (RAS). See *3.15 TBU RAS registers* on page 3-147.

Address offset 0x08E88

Type S, RW

Reset value 1

Bit descriptions

The following figure shows the bit assignments.

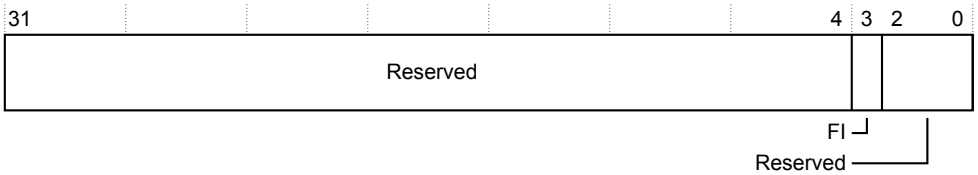

Figure 3-36 TBU_ERRCTLR register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-62 TBU_ERRCTLR register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved
[3]	FI	Fault handling interrupt enable
[2:0]	-	Reserved

3.15.3 TBU_ERRSTATUS register

This section describes the TBU_ERRSTATUS register.

Certain bits in this register are cleared by writing a 1 to their bit position. These writes are ignored in certain circumstances to avoid race conditions where a new error has occurred that software has not yet observed.

Configurations

This register is available in all configurations.

Attributes

The TBU_ERRSTATUS register attributes are as follows:

Width 32-bit

Functional group TBU *Reliability, Availability, and Serviceability* (RAS). See *3.15 TBU RAS registers* on page 3-147.

Address offset `0x08E90`

Type Secure, RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

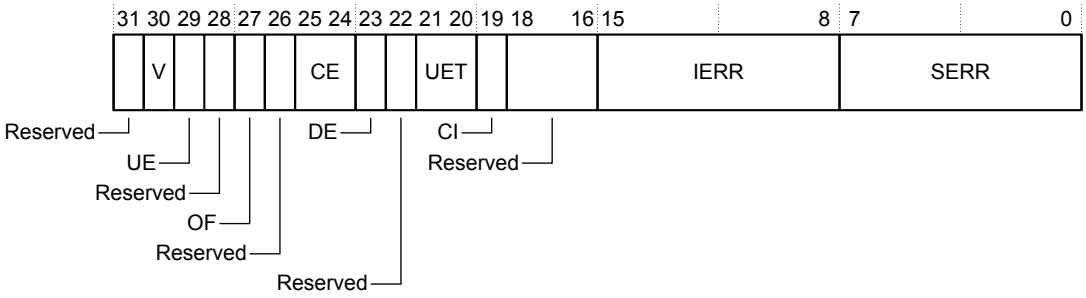

Figure 3-37 TBU_ERRSTATUS register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-63 TBU_ERRSTATUS register bit descriptions

Bits	Name	Description
[31]	-	Reserved
[30]	V	Status Register valid. The possible values of this bit are as follows: `0` ERRSTATUS is not valid `1` ERRSTATUS is valid, meaning that at least one error has been recorded This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See *3.1.1 Clearing ERRSTATUS registers* on page 3-87. This bit resets to zero on a reset.
[29]	UE	Uncorrected errors. The possible values of this bit are: `0` No errors that could not be corrected or deferred `1` At least one error is detected that has not been corrected or deferred This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See *3.1.1 Clearing ERRSTATUS registers* on page 3-87.
[28]	-	Reserved
[27]	OF	Overflow. Multiple errors detected. This bit is set to 1 when: • Any error is received and TBU_ERRSTATUS.V is already set, and not being cleared on the same cycle • Multiple errors are received on the same cycle This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See *3.1.1 Clearing ERRSTATUS registers* on page 3-87.
[26]	-	Reserved
[25:24]	CE	Correctable Errors: `00b` No correctable errors recorded `10b` At least one corrected error recorded Other values are Reserved. Clearing depends on other ERRSTATUS fields. See *3.1.1 Clearing ERRSTATUS registers* on page 3-87.
[23]	DE	Deferred errors. The possible values of this bit are as follows: `0` No errors were deferred `1` At least one error was not corrected and deferred This field is read/write-one-to-clear. Clearing depends on other ERRSTATUS fields. See *3.1.1 Clearing ERRSTATUS registers* on page 3-87.
[22]	-	Reserved
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. The possible values of this field are as follows: `0b00` Uncorrected error, UnContainable error (UC) Writes to this field are ignored.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-63 TBU_ERRSTATUS register bit descriptions (continued)

Bits	Name	Description
[19]	CI	Indicates whether a critical error condition has been recorded. The possible values of this bit are as follows: 0 No critical error condition 1 Critical error condition Writes to this field are ignored.
[18:16]	-	Reserved
[15:8]	IERR	IMPLEMENTATION DEFINED error code. This field indicates the source of the error as follows: 15h BIU WDB ROBUFF_P 14h BIU WDB ROBUFF_C 13h BIU WDB ROBUFF_D 12h Reserved 11h Reserved 10h TLBU DCU MTLB DATA 0Fh TLBU DCU MTLB TAGS 0Eh TLBU DCU MTLB REPL 0Dh TLBU DCU MTLB PCNT 0Ch TLBU DCU MTLB PLIM 0Bh TLBU TOU HLB_ENTRY RIGHT 0Ah TLBU TOU HLB_ENTRY LEFT 09h TLBU TOU HLB PTR RIGHT 08h TLBU TOU HLB PTR LEFT 07h Reserved 06h Reserved 05h TLBU TOU DTIQ 04h TLBU TOU UOQ 03h TLBU TOU OGQ 02h TLBU TOU LB 01h TLBU TOU RSP 00h TLBU TOU REQ. Writes to this field are ignored.
[7:0]	SERR	Error code. This provides information about the earliest unacknowledged Error. It can contain the following values: 0 No error 2 Single or double error from RAMs that are not MTLB TAGS or DATA 8 Single or double error from MTLB Data 9 Single or double error from MTLB Tags All other values are reserved. Writes to this field are ignored.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.15.4 TBU_ERRGEN register

Error Generation Register. Use the TBU_ERRGEN register to test software for when a RAS error occurs.

The field locations are same as for the *3.15.3 TBU_ERRSTATUS register* on page 3-149.

When this register is updated, the *3.15.3 TBU_ERRSTATUS register* on page 3-149 is also updated with the same value, as long as the write data generates a valid error record.

A write to ERRGEN is valid if all the following is true:

- ERRGEN.V is set
- At least one of the following is true (CE is legal if CE == `2'b00` or CE == `2'b10`):
  - ERRGEN.UE is set and CE is legal
  - ERRGEN.DE is set and CE is legal
  - ERRGEN.CE is set to `2'b10`
- UET must be `2'b00`

If a valid error record is written, then the appropriate interrupt, or interrupts, are also generated.

This register has identical fields to *3.15.3 TBU_ERRSTATUS register* on page 3-149.

Configurations

This register is available in all configurations.

Attributes

The TBU_ERRGEN register attributes are as follows:

Width	32-bit
Functional group	TBU *Reliability, Availability, and Serviceability* (RAS). See *3.15 TBU RAS registers* on page 3-147.
Address offset	`0x08EC0`
Type	S, RW
Reset value	`0`

Bit descriptions

See the bit descriptions in *3.15.3 TBU_ERRSTATUS register* on page 3-149.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.16 TBU system discovery registers

This section describes the TBU system discovery registers.

This section contains the following subsections:

3.16.1 TBU_SYSDISC0 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC0 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E30`

Type RO

Reset value `TBUCFG_MTLB_DEPTH`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

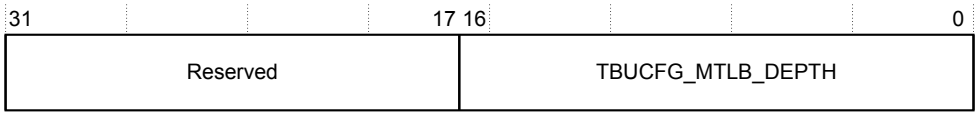

Figure 3-38 TBU_SYSDISC0 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-64 TBU_SYSDISC0 register bit descriptions

Bits	Name	Description
[31:17]	-	Reserved

[16:0]	TBUCFG_MTLB_DEPTH	The read data reflects the chosen parameter value, for example:
		`17'h0_0008 : 8`
		….
		`17'h1_0000 : 65536`

3.16.2 TBU_SYSDISC1 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC1 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E34`

Type RO

Reset value `TBUCFG_UTLB_DEPTH`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

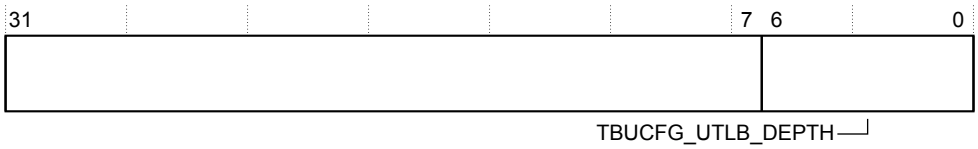

Figure 3-39 TBU_SYSDISC1 register bit assignments

The following table shows the bit descriptions.

Table 3-65 TBU_SYSDISC1 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved

[6:0]	TBUCFG_UTLB_DEPTH	The read data reflects the chosen parameter value, for example:
		`7'h04 : 4`
		….
		`7'h40 : 64`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.16.3 TBU_SYSDISC2 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC2 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E38`

Type RO

Reset value `TBUCFG_MTLB_WAYS`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

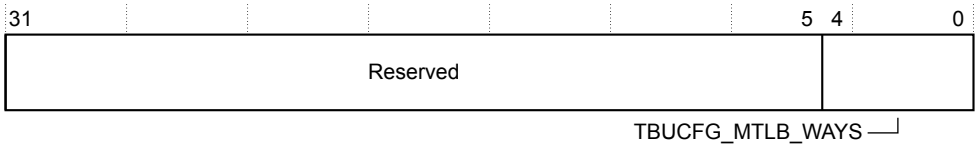

Figure 3-40 TBU_SYSDISC2 register bit assignments

The following table shows the bit descriptions.

Table 3-66 TBU_SYSDISC2 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved

[4:0]	TBUCFG_MTLB_WAYS	The read data reflects the chosen parameter value, for example:
		`5'h04 : 4`
		….
		`5'h10 : 16`

3.16.4 TBU_SYSDISC3 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC3 register attributes are as follows:

Width 32-bit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E3C`

Type RO

Reset value `TBUCFG_MTLB_BANKS`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

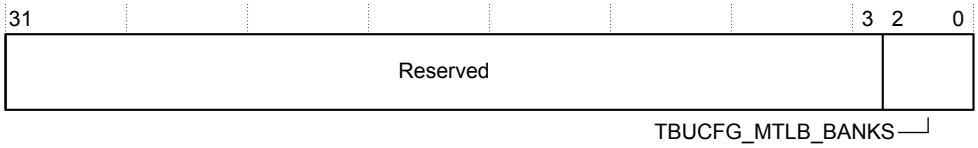

Figure 3-41 TBU_SYSDISC3 register bit assignments

The following table shows the bit descriptions.

Table 3-67 TBU_SYSDISC3 register bit descriptions

Bits	Name	Description
[31:3]	-	Reserved

[2:0]	TBUCFG_MTLB_BANKS	The read data reflects the chosen parameter value, for example:
		`3'b001 : 1`
		….
		`3'b100 : 4`

3.16.5 TBU_SYSDISC4 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC4 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E40`

Type RO

Reset value `TBUCFG_XLATE_SLOTS`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

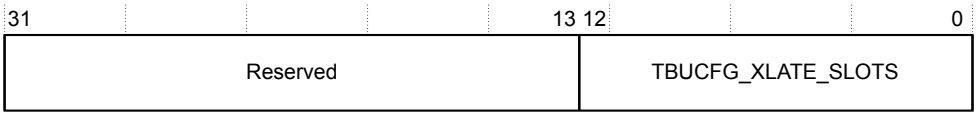

Figure 3-42 TBU_SYSDISC4 register bit assignments

The following table shows the bit descriptions.

Table 3-68 TBU_SYSDISC4 register bit descriptions

Bits	Name	Description
[31:13]	-	Reserved

[12:0]	TBUCFG_XLATE_SLOTS	The read data reflects the chosen parameter value, for example:
		`13'h0004 : 4`
		….
		`13'h1000 : 4096`

3.16.6 TBU_SYSDISC5 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC5 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E44`

Type RO

Reset value `TBUCFG_PMU_COUNTERS`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

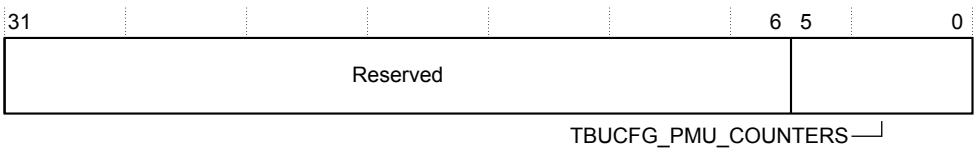

Figure 3-43 TBU_SYSDISC5 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-69 TBU_SYSDISC5 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved

[5:0]	TBUCFG_PMU_COUNTERS	The read data reflects the chosen parameter value, for example:
		`6'h04 : 4`
		….
		`6'h20 : 32`

3.16.7 TBU_SYSDISC6 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC6 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E48`

Type RO

Reset value `TBUCFG_SID_WIDTH`. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.

Bit descriptions

The following figure shows the bit assignments.

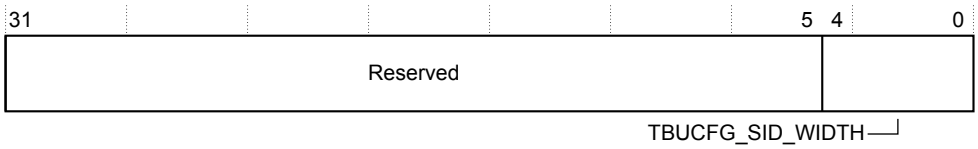

Figure 3-44 TBU_SYSDISC6 register bit assignments

The following table shows the bit descriptions.

Table 3-70 TBU_SYSDISC6 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved

[4:0]	TBUCFG_SID_WIDTH	The read data reflects the chosen parameter value, for example:
		`5'h08 : 8`
		….
		`5'h14 : 20`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.16.8 TBU_SYSDISC7 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC7 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E4C`

Type RO

Reset value `TBUCFG_SSID_WIDTH`. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.

Bit descriptions

The following figure shows the bit assignments.

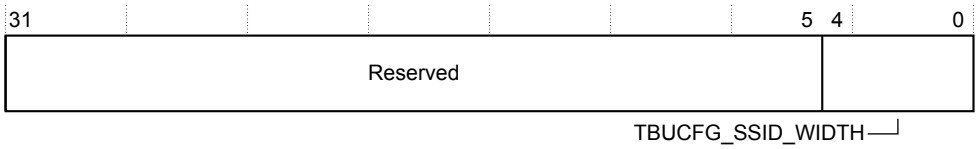

Figure 3-45 TBU_SYSDISC7 register bit assignments

The following table shows the bit descriptions.

Table 3-71 TBU_SYSDISC7 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved

[4:0]	TBUCFG_SSID_WIDTH	The read data reflects the chosen parameter value, for example:
		`5'h01 : 1`
		….
		`5'h14 : 20`

3.16.9 TBU_SYSDISC8 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC8 register attributes are as follows:

Width 32-bit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E50`

Type RO

Reset value `TBUCFG_DIRECT_IDX`. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.

Bit descriptions

The following figure shows the bit assignments.

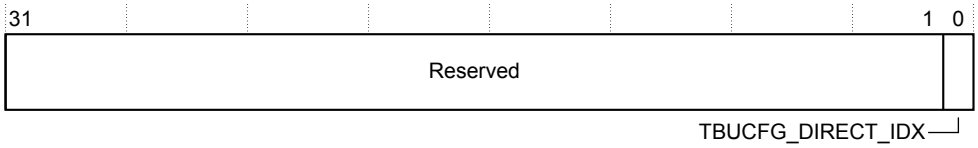

Figure 3-46 TBU_SYSDISC8 register bit assignments

The following table shows the bit descriptions.

Table 3-72 TBU_SYSDISC8 register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved

[0]	TBUCFG_DIRECT_IDX	The read data reflects the chosen parameter value, for example:
		`1'b0 : 0`
		….
		`1'b1 : 1`

3.16.10 TBU_SYSDISC9 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC9 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E54`

Type RO

Reset value `TBUCFG_MTLB_PARTS`. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

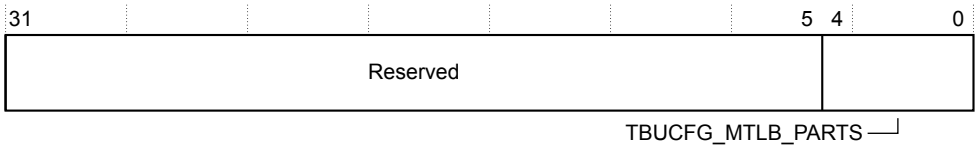

Figure 3-47 TBU_SYSDISC9 register bit assignments

The following table shows the bit descriptions.

Table 3-73 TBU_SYSDISC9 register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved

[4:0]	TBUCFG_MTLB_PARTS	The read data reflects the chosen parameter value, for example:
		`5'h00 : 0`
		….
		`5'h10 : 16`

3.16.11 TBU_SYSDISC10 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC10 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E58`

Type RO

Reset value `TBUCFG_LTI_OG_WIDTH`. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.

Bit descriptions

The following figure shows the bit assignments.

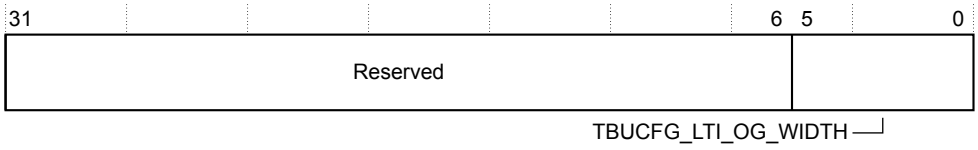

Figure 3-48 TBU_SYSDISC10 register bit assignments

The following table shows the bit descriptions.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table 3-74 TBU_SYSDISC10 register bit descriptions

Bits	Name	Description
[31:6]	-	Reserved

[5:0]	TBUCFG_LTI_OG_WIDTH	The read data reflects the chosen parameter value, for example:
		`6'h00 : 0`
		….
		`6'h20 : 32`

3.16.12 TBU_SYSDISC11 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC11 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E5C`

Type RO

Reset value `TBUCFG_PARTID_WIDTH`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

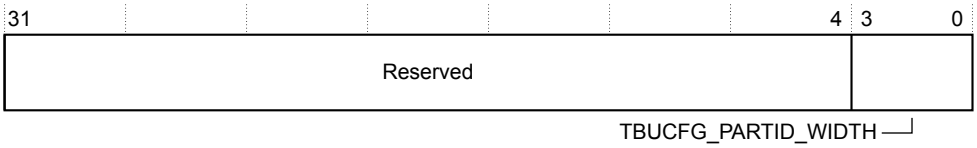

Figure 3-49 TBU_SYSDISC11 register bit assignments

The following table shows the bit descriptions.

Table 3-75 TBU_SYSDISC11 register bit descriptions

Bits	Name	Description
[31:4]	-	Reserved

[3:0]	TBUCFG_PARTID_WIDTH	The read data reflects the chosen parameter value, for example:
		`4'b0001 : 1`
		….
		`4'b1001 : 9`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.16.13 TBU_SYSDISC12 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC12 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E60`

Type RO

Reset value `TBUCFG_HZRD_ENTRIES`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

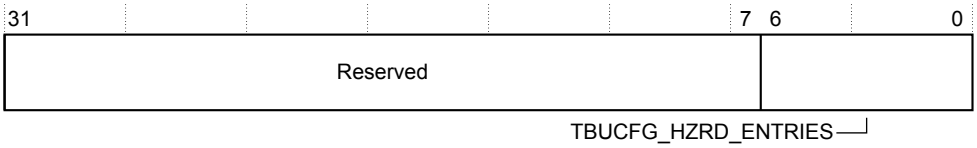

Figure 3-50 TBU_SYSDISC12 register bit assignments

The following table shows the bit descriptions.

Table 3-76 TBU_SYSDISC12 register bit descriptions

Bits	Name	Description
[31:7]	-	Reserved

[6:0]	TBUCFG_HZRD_ENTRIES	The read data reflects the chosen parameter value, for example:
		`7'h00 : 0`
		….
		`7'h40 : 64`

3.16.14 TBU_SYSDISC13 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC13 register attributes are as follows:

Width 32-bit

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E64`

Type RO

Reset value `TBUCFG_SLOTRAM_TYPE`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

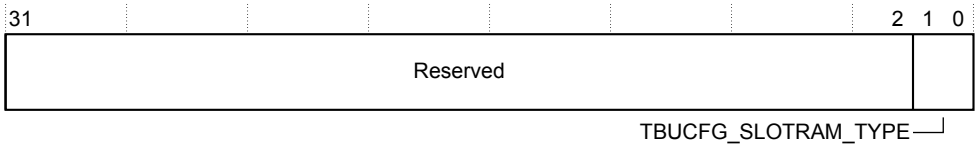

Figure 3-51 TBU_SYSDISC13 register bit assignments

The following table shows the bit descriptions.

Table 3-77 TBU_SYSDISC13 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TBUCFG_SLOTRAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2'b00 : 0`
		….
		`2'b10 : 2`

3.16.15 TBU_SYSDISC14 system discovery register

The TBU system discovery registers discover components in the system.

Configurations

This register is available in all configurations.

Attributes

The TBU_SYSDISC14 register attributes are as follows:

Width 32-bit

Functional group TBU system discovery registers. See *3.16 TBU system discovery registers* on page 3-153.

Address offset `0x08E68`

Type RO

Reset value `TBUCFG_CACHERAM_TYPE`. See *2.5.8 TBU buffer configuration options* on page 2-81.

Bit descriptions

The following figure shows the bit assignments.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

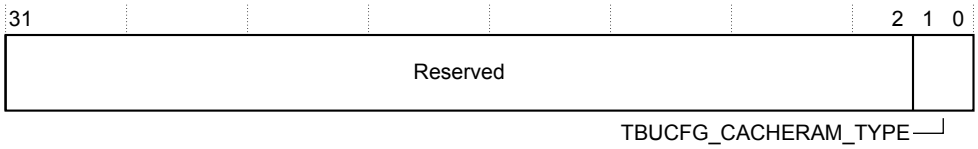

Figure 3-52 TBU_SYSDISC14 register bit assignments

The following table shows the bit descriptions.

Table 3-78 TBU_SYSDISC14 register bit descriptions

Bits	Name	Description
[31:2]	-	Reserved

[1:0]	TBUCFG_CACHERAM_TYPE	The read data reflects the chosen parameter value, for example:
		`2'b00 : 0`
		….
		`2'b01 : 1`

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

3.17 TBU integration registers

This section describes the TBU integration registers.

This section contains the following subsections:

- *3.17.1 ITEN register* on page 3-166.
- *3.17.2 ITOP_TBU register* on page 3-166.
- *3.17.3 ITIN_TBU register* on page 3-167.

3.17.1 ITEN register

This section describes the ITEN register.

Configurations

This register is available in all configurations.

Attributes

The ITEN register attributes are as follows:

Width 32-bit

Functional group TBU integration registers. See *3.4.11 TBU integration registers summary* on page 3-100.

Address offset `0x08E20`

Type RW

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

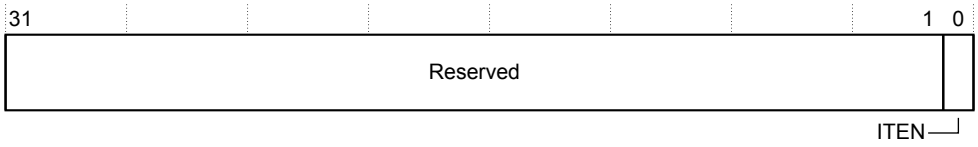

Figure 3-53 ITEN register bit assignments

The following table shows the bit descriptions.

Table 3-79 ITEN register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	ITEN	`0` Integration mode is disabled `1` Integration mode is enabled

3.17.2 ITOP_TBU register

This section describes the ITOP register for the TBU.

Configurations

This register is available in all configurations.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Attributes

The ITOP_TBU register attributes are as follows:

Width 32-bit

Functional group TBU integration registers. See *3.4.11 TBU integration registers summary* on page 3-100.

Address offset `0x08E24`

Type RW

Reset value `0`

Bit descriptions

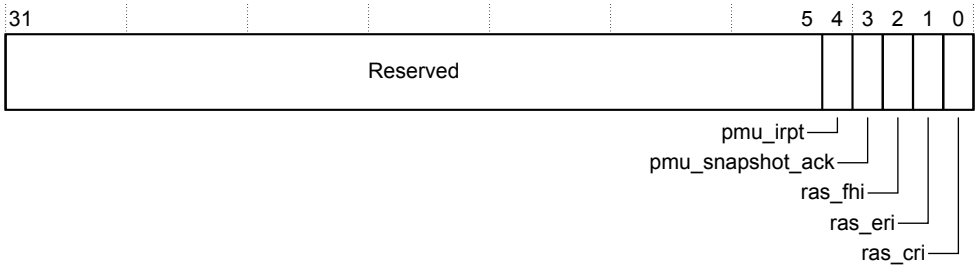

Figure 3-54 ITOP_TBU register bit assignments

The following table shows the bit descriptions.

Table 3-80 ITOP_TBU register bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, SBZ
[4]	pmu_irpt	`0` When ITEN.ITEN == 0, SBZ `1` When ITEN.ITEN == 1, the value driven to **pmu_irpt**
[3]	pmu_snapshot_ack	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **pmusnapshot_ack**
[2]	ras_fhi	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **ras_fhi**
[1]	ras_eri	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **ras_eri**
[0]	ras_cri	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, the value driven to **ras_cri**

See:

- *A.2.8 TBU interrupt signals* on page Appx-A-192
- *A.1.5 TCU PMU snapshot interface signals* on page Appx-A-174

3.17.3 ITIN_TBU register

This section describes the ITIN register for the TBU.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Configurations

This register is available in all configurations.

Attributes

The ITIN_TBU register attributes are as follows:

Width 32-bit

Functional group TBU integration registers. See *3.4.11 TBU integration registers summary* on page 3-100.

Address offset `0x08E28`

Type RO

Reset value `0`

Bit descriptions

The following figure shows the bit assignments.

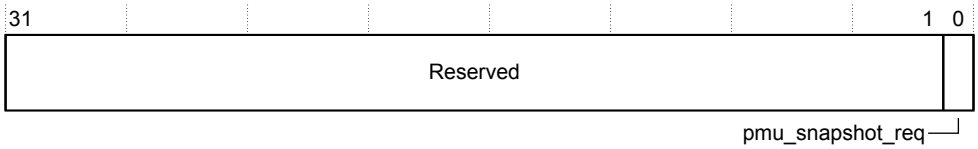

Figure 3-55 ITIN_TBU register bit assignments

The following table shows the bit descriptions.

Table 3-81 ITIN_TBU register bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, SBZ
[0]	pmu_snapshot_req	`0` When ITEN.ITEN == `0`, SBZ `1` When ITEN.ITEN == `1`, reflects **pmusnapshot_req**

See *A.1.5 TCU PMU snapshot interface signals* on page Appx-A-174.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Appendix A
Signal descriptions

This appendix describes the MMU-700 external signals.

It contains the following sections:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.1 TCU signals

This section describes the MMU-700 TCU signals.

This section contains the following subsections:

A.1.1 TCU clock and reset signals

The TCU uses a single set of standard clock and reset signals.

The following table shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Width	Direction	Description
clk	1-bit	Input	Global clock
resetn	1-bit	Input	Global reset

A.1.2 TCU QTW/DVM interface signals

The TCU QTW/DVM interface signals are based on the AMBA ACE5-Lite signals.

The following table shows the TCU QTW/DVM interface signals.

Table A-2 TCU QTW/DVM interface signals

Signal	Width	Direction	Description
acaddr_qtw	52-bit	Input	Snoop address
acprot_qtw	3-bit	Input	Snoop protection type
acready_qtw	1-bit	Output	Snoop address ready
acsnoop_qtw	4-bit	Input	Snoop transaction type
acvalid_qtw	1-bit	Input	Snoop address valid
arid_qtw	See [e]	Output	Read address ID

[e] `QTW_ID_WIDTH`-bit.

`QTW_ID_WIDTH` is calculated as follows:

$((\log_2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) > 4)\ ?\ (\log^2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) : 4$; See *2.5.2 TCU buffer configuration options* on page 2-75.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-2 TCU QTW/DVM interface signals (continued)

Signal	Width	Direction	Description
araddr_qtw	52-bit	Output	Read address
arburst_qtw	2-bit	Output	Burst type
arcache_qtw	4-bit	Output	Memory type
ardomain_qtw	2-bit	Output	Shareability domain
arlen_qtw	8-bit	Output	Burst length
arlock_qtw	1-bit	Output	Lock type
arprot_qtw	3-bit	Output	Protection type
arqos_qtw	4-bit	Output	QoS identifier
arready_qtw	1-bit	Input	Read address ready
arsize_qtw	3-bit	Output	Burst size
arsnoop_qtw	4-bit	Output	Transaction type
arvalid_qtw	1-bit	Output	Read address valid
awid_qtw	See [f]	Output	Write address ID
awaddr_qtw	52-bit	Output	Write address
awburst_qtw	2-bit	Output	Burst type
awcache_qtw	4-bit	Output	Memory type
awdomain_qtw	2-bit	Output	Shareability domain
awlen_qtw	8-bit	Output	Burst length
awlock_qtw	1-bit	Output	Lock type
awprot_qtw	3-bit	Output	Protection type
awqos_qtw	4-bit	Output	QoS identifier
awready_qtw	1-bit	Input	Write address ready
awsize_qtw	3-bit	Output	Burst size
awsnoop_qtw	4-bit	Output	Transaction type
awvalid_qtw	1-bit	Output	Write address valid

[f] `QTW_ID_WIDTH`-bit.

`QTW_ID_WIDTH` is calculated as follows:

$((\log_2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) > 4) \; ? \; (\log^2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) : 4$; See *2.5.2 TCU buffer configuration options* on page 2-75.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-2 TCU QTW/DVM interface signals (continued)

Signal	Width	Direction	Description
crready_qtw	1-bit	Input	Snoop response ready
crresp_qtw	5-bit	Output	Snoop response
crvalid_qtw	1-bit	Output	Snoop response valid
rid_qtw	See [g]	Input	Read data ID
rdata_qtw	See [h]	Input	Read data
rlast_qtw	1-bit	Input	Read last
rready_qtw	1-bit	Output	Read ready
rresp_qtw	2-bit	Input	Read response
rvalid_qtw	1-bit	Input	Read valid
wdata_qtw	See [i]	Output	Write data
wlast_qtw	1-bit	Output	Write last
wready_qtw	1-bit	Input	Write ready
wstrb_qtw	See [j]	Output	Write strobe
wvalid_qtw	1-bit	Output	Write valid
bid_qtw	See [k]	Input	Response ID
bready_qtw	1-bit	Output	Response ready
bresp_qtw	2-bit	Input	Write response
bvalid_qtw	1-bit	Input	Write response valid
awakeup_qtw	1-bit	Output	Wakeup
acwakeup_qtw	1-bit	Input	Snoop wakeup
acvmidext_qtw	4-bit	Input	Snoop Extended *Virtual Machine IDentifier* (VMID)

For more information about these signals, see the *AMBA® AXI and ACE Protocol Specification*.

g `QTW_ID_WIDTH`-bit.

`QTW_ID_WIDTH` is calculated as follows:

$((\log_2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) > 4)\ ?\ (\log^2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) : 4$; See *2.5.2 TCU buffer configuration options* on page 2-75.

h `TCUCFG_QTW_DATA_WIDTH`-bit. See *2.5.1 TCU I/O configuration options* on page 2-75.

i `TCUCFG_QTW_DATA_WIDTH`-bit. See *2.5.1 TCU I/O configuration options* on page 2-75.

j (`TCUCFG_QTW_DATA_WIDTH`/8)-bit. See *2.5.1 TCU I/O configuration options* on page 2-75.

k `QTW_ID_WIDTH`-bit.

`QTW_ID_WIDTH` is calculated as follows:

$((\log_2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) > 4)\ ?\ (\log^2(\texttt{TCUCFG\_PTW\_SLOTS}) + 2) : 4$; See *2.5.2 TCU buffer configuration options* on page 2-75.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.1.3 TCU programming interface signals

The TCU programming interface signals are based on the AMBA APB4 signals.

The following table shows the TCU programming interface signals.

Table A-3 TCU programming interface signals

Signal	Width	Direction	Description
paddr_prog	See [1]	Input	Peripheral address
psel_prog	1-bit	Input	Peripheral select
penable_prog	1-bit	Input	Enable for transfer
pwrite_prog	1-bit	Input	Write transaction indicator
pprot_prog	3-bit	Input	Protection type
pwdata_prog	32-bit	Input	Write data
pstrb_prog	4-bit	Input	Write data strobe
pslverr_prog	1-bit	Output	Error response
prdata_prog	32-bit	Output	Read data
pready_prog	1-bit	Output	Transfer ready
pwakeup_prog	1-bit	Input	Interface wakeup

For more information about these signals, see the *AMBA® APB Protocol Specification*.

A.1.4 TCU SYSCO interface signals

The following table shows the TCU SYSCO interface signals.

Table A-4 TCU SYSCO interface signals

Signal	Width	Direction	Description
syscoreq_qtw	1-bit	Output	System coherency request. This output transitions: **HIGH** To indicate that the master is requesting to enter the coherency domain. **LOW** To indicate that the master is requesting to exit the coherency domain.
syscoack_qtw	1-bit	Input	System coherency acknowledge. This input transitions to the same level as **syscoreq_qtw** when the request to enter or exit the coherency domain is complete.

For more information about these signals, see the *AMBA® AXI and ACE Protocol Specification*.

[1] If TCUCFG_NUM_TBU is 62, the width of **paddr_prog** is 23-bit. Otherwise, the width of **paddr_prog** is 21-bit. See *2.5.2 TCU buffer configuration options on page 2-75*.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.1.5 TCU PMU snapshot interface signals

The following table shows the TCU PMU snapshot interface signals.

Table A-5 TCU PMU snapshot interface signals

Signal	Width	Direction	Description
pmusnapshot_req	1-bit	Input	PMU snapshot request. The PMU snapshot occurs on the rising edge of **pmusnapshot_req**. ——— **Note** ——— Connect to the debug infrastructure of your SoC. ———
pmusnapshot_ack	1-bit	Output	PMU snapshot acknowledge. The TCU uses this signal to acknowledge that the PMU snapshot has occurred. This signal is LOW after reset. ——— **Note** ——— Connect to the debug infrastructure of your SoC. ———

A.1.6 TCU LPI_PD interface signals

The following table shows the TCU LPI_PD interface signals.

Table A-6 TCU LPI_PD interface signals

Signal	Width	Direction	Description
qactive_pd	1-bit	Output	Component active
qreqn_pd	1-bit	Input	Quiescence request
qacceptn_pd	1-bit	Output	Quiescence accept
qdeny_pd	1-bit	Output	Quiescence deny

For more information about these signals, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

A.1.7 TCU LPI_CG interface signals

The following table shows the TCU LPI_CG interface signals.

Table A-7 TCU LPI_CG interface signals

Signal	Width	Direction	Description
qactive_cg	1-bit	Output	Component active
qreqn_cg	1-bit	Input	Quiescence request
qacceptn_cg	1-bit	Output	Quiescence accept
qdeny_cg	1-bit	Output	Quiescence deny

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

For more information about these signals, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

A.1.8 TCU DTI interface signals

The following table shows the TCU DTI interface signals.

Table A-8 TCU DTI interface signals

Signal	Width	Direction	Description
tvalid_dti_dn	1-bit	Master to slave	Flow control signal
tready_dti_dn	1-bit	Slave to master	Flow control signal
tdata_dti_dn	20-bit	Master to slave	Message data signal
tid_dti_dn	See [m]	Master to slave	Identifies the master that initiated the message
tlast_dti_dn	1-bit	Master to slave	Indicates the last cycle of a message
tkeep_dti_dn	20-bit	Master to slave	This signal indicates valid bytes
tvalid_dti_up	1-bit	Slave to master	Flow control signal
tready_dti_up	1-bit	Master to slave	Flow control signal
tdata_dti_up	160-bit	Slave to master	Message data signal
tdest_dti_up	See [n]	Slave to master	Identifies the master that is receiving the message
tlast_dti_up	1-bit	Slave to master	Indicates the last cycle of a message
tkeep_dti_up	20-bit	Slave to master	Indicates valid bytes
twakeup_dti_up	1-bit	Slave to master	Wakeup signal
twakeup_dti_dn	1-bit	Master to slave	Wakeup signal

For more information about the DTI signals, see the *AMBA® 4 AXI4-Stream Protocol Specification*.

For more information about DTI protocol messages, see the *AMBA® DTI Protocol Specification*.

A.1.9 TCU interrupt signals

The TCU interrupt signals are edge-triggered. The interrupt controller must detect the rising edge of these signals.

The TCU can also output the Secure and Non-secure Event queue, SYNC complete commands, and global interrupts as *Message Signaled Interrupts* (MSIs) on the QTW/DVM interface. If the system supports capturing MSIs from the TCU, there is no requirement to connect the corresponding interrupt signals in this interface.

The following table shows the TCU interrupt signals.

[m] If `TCUCFG_NUM_TBU` is 62, the width of **tid_dti_dn** is 6-bit. Otherise, the width of **tid_dti_dn** is 4-bit. See *2.5.2 TCU buffer configuration options* on page 2-75.
[n] If `TCUCFG_NUM_TBU` is 62, the width of **tdest_dti_up** is 6-bit. Otherise, the width of **tdest_dti_up** is 4-bit. See *2.5.2 TCU buffer configuration options* on page 2-75.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-9 TCU interrupt interface signals

Signal	Width	Direction	Description
event_q_irpt_s	1-bit	Output	Event queue, Secure interrupt. Asserts a Secure interrupt to indicate that the Event queue is not empty or has overflowed.
event_q_irpt_ns	1-bit	Output	Event queue, Non-secure interrupt. Asserts a Non-secure interrupt to indicate that the Event queue is not empty or has overflowed.
cmd_sync_irpt_ns	1-bit	Output	SYNC complete, Non-secure interrupt. Asserts a Non-secure interrupt to indicate that the `CMD_SYNC` command is complete.
cmd_sync_irpt_s	1-bit	Output	SYNC complete, Secure interrupt. Asserts a Secure interrupt to indicate that the `CMD_SYNC` command is complete.
global_irpt_ns	1-bit	Output	Asserts a global Non-secure interrupt
global_irpt_s	1-bit	Output	Asserts a global Secure interrupt
ras_fhi	1-bit	Output	Fault handling RAS interrupt for a contained error
ras_eri	1-bit	Output	Error recovery RAS interrupt for an uncontained error
ras_cri	1-bit	Output	Critical error interrupt, for an uncontainable uncorrected error
pmu_irpt	1-bit	Output	Asserts a PMU interrupt. **Note** The MMU-700 cannot output PMU interrupts as MSIs. You must connect this output to an interrupt controller.
pri_q_irpt_ns	1-bit	Output	Asserts a *Page Request Interface* (PRI) queue interrupt

A.1.10 TCU MSI interface signals

This section describes the TCU *Message Signaled Interrupt* (MSI) interface.

See the *GIC MSI Delivery Interface* document for more information.

The interface follows the AXI4-Stream protocol and uses the signals in the following table to send MSIs.

The following table shows the TCU MSI interface signals.

Table A-10 TCU MSI interface signals

Signal	AXI4-Stream signal	Width	Direction	Description
msitvalid	**TVALID**	1-bit	Output	Indicates valid data to the GIC
msitready	**TREADY**	1-bit	Input	Indicates acceptance by the GIC
msitdata	**TDATA**	64-bit	Output	Data being passed to the GIC
msitwakeup	**TWAKEUP**, AMBA extension	1-bit	Output	Indicates that a transaction is ongoing
msirtvalid	**TVALID**	1-bit	Input	Indicates that the GIC has accepted an MSI
msirtready	**TREADY**	1-bit	Output	Indicates that the device has accepted the response packet
msirtwakeup	**TWAKEUP**, AMBA extension	1-bit	Input	Indicates that a transaction is ongoing

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.1.11 TCU event interface signal

The TCU event interface signal is an event output for connection to processors.

The following table shows the TCU event interface signal.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-11 TCU event interface signal

Signal	Width	Direction	Description
evento	1-bit	Output	The **evento** signal is asserted for one cycle to indicate an event that enables processors to wake up from the *Wait For Event* (WFE) low-power state. Connect the **evento** signal of the TCU to the event interface of Arm processors. Processors that use the *DynamIQ Shared Unit* (DSU) have a different event handshake mechanism. The mechanism that the DSU uses is the successor to the mechanism that some MMUs use. Arm processors can use the following event mechanisms: • Some processors have an **eventi** input to connect directly to the **evento** output from the MMU. • Some processors, including DSU-based systems, have a **req**/**ack** handshake mechanism that requires the **evento** signal from the MMU to be converted and uses the **eventiack**, **eventireq**, **eventoack**, and **eventoreq** signals. **Note** You can also route the **evento** signal through other interconnects such as the Arm CoreLink CMN-600 Coherent Mesh Network instead of connecting **evento** directly to the processor. These interconnects, like the DSU, only support the newer event mechanism. If the rest of your system uses the newer event mechanism, you must add logic to convert events that the MMU-700 generates, which uses the older event mechanism. In both mechanisms, in the signal names: **i** Represents events that are inputs to a particular component. **o** Represents events that are outputs from a particular component. **Note** For the signals, the handshake mechanism uses one input and one output in each direction. This is because the acknowledgment of the request operates in the opposite direction to the original request. The MMU-700 has an event output and therefore only has the **evento** signal. The processor has an input interface to receive the event from the MMU-700, and other devices. This input interface uses the **eventiack** and **eventireq** signals, if the processor uses the newer mechanism. The required conversion is from the older mechanism, **eventi** and **evento** signals, to the newer mechanism, **eventiack**, **eventireq**, **eventoack**, and **eventoreq** signals. When connecting the MMU-700 to a DSU, the only signals to consider are the following: • **evento** signal of the MMU-700. • **eventiack** and **eventireq** signals of the DSU. Some processors have an **eventi** input instead. You can use the *Channel Pulse to Channel adapter* that is provided in the CoreSight System-on-Chip SoC-600. For more information about this component, see *Chapter 6.11* in the *Arm® CoreSight™ System-on-Chip SoC-600 Technical Reference Manual*. **Note** To use the *Channel Pulse to Channel adapter* from CoreSight System-on-Chip SoC-600, you must be a licensee of the SoC-600 product. If you are not a licensee of SoC-600, you must add your own logic.

For more information, see the documentation for your processor or DSU.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.1.12 TCU tie-off signals

The TCU tie-off signals are sampled between exiting reset and the LPI_PD interface first entering the Q_RUN state. Ensure that the value of these signals does not change when the LPI_PD interface is in the Q_STOPPED or Q_EXIT state for the first time after exiting reset.

The following table shows the TCU tie-off signals.

Table A-12 TCU tie-off signals

Signal	Width	Direction	Description
sup_cohacc	1-bit	Input	This signal indicates whether the QTW interface is I/O-coherent. Tie HIGH when the TCU is connected to a coherent interconnect.
sup_btm	1-bit	Input	This signal indicates whether the Broadcast TLB Maintenance is supported. Tie HIGH when the TCU is connected to an interconnect that supports DVM.
sup_sev	1-bit	Input	This signal indicates whether the Send Event mechanism is supported. Tie HIGH when **evento** is connected.
sup_oas[2:0]	3-bit	Input	Output address size supported. The encodings for this input are as follows: `0b000` 32 bits `0b001` 36 bits `0b010` 40 bits `0b011` 42 bits `0b100` 44 bits `0b101` 48 bits `0b110` 52 bits You must not use other encodings. Other encodings are treated as `0b110`.
sec_override	1-bit	Input	When HIGH, certain registers are accessible to Non-secure accesses from reset, as the *3.7.7 TCU_SCR register* on page 3-112 settings describe
ecorevnum[3:0]	4-bit	Input	Tie this signal to 0 unless directed otherwise by Arm
msi_addr[51:0]	52-bit	Input	If the programmed MSI address in SMMU_(S_)_*_IRQ_CFG0.ADDR matched **msi_addr**, then an MSI is generated on the GIC AXI-Stream interface

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-12 TCU tie-off signals (continued)

Signal	Width	Direction	Description
tcu_sid[31:0]	32-bit	Input	Used to generate DeviceID for TCU-generated MSIs
sup_httu	1-bit	Input	`0` When set to 0, **sup_httu** indicates that the ACE-Lite interface that is connected to the TCU cannot support atomics. The TCU cannot perform *Hardware Translation Table Update* (HTTU) transactions. `1` When set to 1, **sup_httu** indicates that the ACE-Lite interface that is connected to the TCU can support atomics. The TCU uses atomic transactions to perform HTTU. The impact of **sup_httu** on SMMU_IDR0.HTTU is as follows: **sup_httu is `1'b0`** SMMU_IDR0.HTTU is `2'b00` **sup_httu is `1'b1`** SMMU_IDR0.HTTU is `2'b10` See *2.4.1 SMMUv3 implementation* on page 2-57.

For more information about the SMMUv3 ID signals, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

A.1.13 TCU ELA debug signals

The MMU-700 TCU includes *Embedded Logic Analyzer* (ELA) debug signals.

The following table shows the ELA enable signal.

Table A-13 ELA enable signal

Signal	Width	Direction	Description
ela_enable	1-bit	Input	**ela_enable** is an asynchronous input port. When `TCUCFG_USE_ELA_DEBUG` is 0, the SMMU ignores the value of the signal. When `TCUCFG_USE_ELA_DEBUG` is 1, **ela_enable** acts as a clock enable for the TCU ELA observation interface. If ELA debug is required, drive **ela_enable** HIGH. If ELA debug is not required, drive **ela_enable** LOW to reduce the dynamic power consumption of the SMMU.

The following table shows the TCU ELA debug signals.

Table A-14 TCU ELA debug signals

Signal	Width	Direction	Description
Signal group 0 signals			
signalgrp0	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual0	4-bit		
sigclken0	1-bit		
Signal group 1 signals			
signalgrp1	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual1	4-bit		
sigclken1	1-bit		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-14 TCU ELA debug signals (continued)

Signal	Width	Direction	Description
Signal group 2 signals			
signalgrp2	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual2	4-bit		
sigclken2	1-bit		
Signal group 3 signals			
signalgrp3	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual3	4-bit		
sigclken3	1-bit		
Signal group 4 signals			
signalgrp4	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual4	4-bit		
sigclken4	1-bit		
Signal group 5 signals			
signalgrp5	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual5	4-bit		
sigclken5	1-bit		
Signal group 6 signals			
signalgrp6	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual6	4-bit		
sigclken6	1-bit		
Signal group 7 signals			
signalgrp7	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual7	4-bit		
sigclken7	1-bit		
Signal group 8 signals			
signalgrp8	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual8	4-bit		
sigclken8	1-bit		
Signal group 9 signals			
signalgrp9	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual9	4-bit		
sigclken9	1-bit		
Signal group 10 signals			

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-14 TCU ELA debug signals (continued)

Signal	Width	Direction	Description
signalgrp10	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual10	4-bit		
sigclken10	1-bit		
Signal group 11 signals			
signalgrp11	128-bit	Output	See *B.1 TCU observation interfaces* on page Appx-B-204
sigqual11	4-bit		
sigclken11	1-bit		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.2 TBU signals

This section describes the MMU-700 TBU signals.

This section contains the following subsections:

A.2.1 TBU clock and reset signals

The TBU uses a single set of standard clock and reset signals.

The following table shows the clock and reset signals.

Table A-15 Clock and reset signals

Signal	Width	Direction	Description
clk	1-bit	Input	Global clock
resetn	1-bit	Input	Global reset

A.2.2 TBU TBS interface signals

The TBU TBS interface signals are based on the AMBA ACE5-Lite signals.

The following table shows the TBU TBS interface signals.

Table A-16 TBU TBS interface signals

Signal	Width	Direction	Description
clk	1-bit	Input	Clock input
resetn	1-bit	Input	Active-LOW reset signal
araddr_s	64-bit	Input	Read address
arburst_s	2-bit	Input	Burst type
arcache_s	4-bit	Input	Memory type
ardomain_s	2-bit	Input	Shareability domain
arid_s	See [o]	Input	Read address ID
arlen_s	8-bit	Input	Burst length
arlock_s	1-bit	Input	Lock type

[o] `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-16 TBU TBS interface signals (continued)

Signal	Width	Direction	Description
arprot_s	3-bit	Input	Protection type
arqos_s	4-bit	Input	*Quality of Service* (QoS)
arready_s	1-bit	Output	Read address ready
arregion_s	4-bit	Input	Region identifier
arsize_s	3-bit	Input	Burst size
armmussid_s	See [p]	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. These signals are defined by the AXI5 Untranslated_Transactions extension.
armmusid_s	See [q]	Input	
armmussidv_s	1-bit	Input	
armmusecsid_s	1-bit	Input	
arvalid_s	1-bit	Input	Read address valid
awaddr_s	64-bit	Input	Write address
awatop_s	6-bit	Input	Atomic operation
awburst_s	2-bit	Input	Burst type
awcache_s	4-bit	Input	Memory type
awdomain_s	4-bit	Input	Shareability domain
awid_s	See [r]	Input	Write address ID
awlen_s	8-bit	Input	Burst length
awlock_s	1-bit	Input	Lock type
awprot_s	3-bit	Input	Protection type
awqos_s	4-bit	Input	QoS
awready_s	1-bit	Output	Write address ready
awregion_s	4-bit	Input	Region identifier
awsize_s	3-bit	Input	Burst size

[p] `TBUCFG_SSID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
[q] `TBUCFG_SID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
[r] `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-16 TBU TBS interface signals (continued)

Signal	Width	Direction	Description
awmmussid_s	See [s]	Input	These signals indicate the StreamID, SubstreamID, and ATS translated status of the originating transaction. These signals are defined by the AXI5 Untranslated_Transactions extension.
awmmusid_s	See [t]		
awmmussidv_s	1-bit		
awmmusecsid_s	1-bit		
awvalid_s	1-bit	Input	Write address valid
bid_s	See [u]	Output	Response ID
bready_s	1-bit	Input	Response ready
bresp_s	3-bit	Output	Write response
bvalid_s	1-bit	Output	Write response valid
rdata_s	See [v]	Output	Read data
rid_s	See [w]	Output	Read ID
rlast_s	1-bit	Output	Read last
rready_s	1-bit	Input	Read ready
rresp_s	3-bit	Output	Read response
rvalid_s	1-bit	Output	Read valid
wdata_s	See [x]	Input	Write data
wlast_s	1-bit	Input	Write last
wready_s	1-bit	Output	Write ready
wstrb_s	See [y]	Input	Write strobes
wvalid_s	1-bit	Input	Write valid
aruser_s	See [z]	Input	Read address (AR) channel User signal
awuser_s	See [aa]	Input	Write address (AW) channel User signal
wuser_s	See [ab]	Input	Write data (W) channel User signal

s `TBUCFG_SSID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
t `TBUCFG_SID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
u `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
v `TBUCFG_DATA_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
w `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
x `TBUCFG_DATA_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
y (`TBUCFG_DATA_WIDTH`/8)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
z (`TBUCFG_ARUSER_WIDTH` + `LTI_TLBLOC_WIDTH_RAW` -)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
aa (`TBUCFG_AWUSER_WIDTH` + `LTI_TLBLOC_WIDTH_RAW` -)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
ab `TBUCFG_WUSER_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-16 TBU TBS interface signals (continued)

Signal	Width	Direction	Description
ruser_s	See [ac]	Output	Read data (R) channel User signal
buser_s	See [ad]	Output	Write response (B) channel User signal
awakeup_s	1-bit	Input	Wakeup signal
arsnoop_s	4-bit	Input	Transaction type of read transaction
awsnoop_s[3]	4-bit	Input	Transaction type of write transaction
awstashnid_s[10:0]	11-bit	Input	These signals are defined by the AXI5 Cache_Stash_Transactions extension. If TBUCFG_STASH = 0, these signals are ignored.
awstashniden_s	1-bit	Input	
awstashlpid_s[4:0]	5-bit	Input	
awstashlpiden_s	1-bit	Input	
archunken_s	1-bit	Input	Read data chunking enable
aridunq_s	1-bit	Input	Read address channel unique ID indicator, active-HIGH
arloop_s	See [ae]	Input	Loopback value for a read transaction. Reflected back on **RLOOP**.
awidunq_s	1-bit	Input	Write address channel unique ID indicator, active-HIGH
awloop_s	See [af]	Input	Loopback value for a write transaction
wpoison_s	See [ag]	Input	Indicates that the write data in this transfer has been corrupted
bidunq_s	1-bit	Input	Write response channel unique ID indicator, active-HIGH
bloop_s	See [ah]	Input	Loopback value for a write response
rchunknum_s	[CHUNKNUM_WIDTH - 1:0]	Input	Read data chunk number
rchunkstrb_s	[CHUNKSTRB_WIDTH - 1:0]	Input	Read data chunk strobe
rchunkv_s	1-bit	Input	Valid signal of **RCHUNKNUM** and **RCHUNKSTRB**
ridunq_s	1-bit	Input	Read data channel unique ID indicator, active-HIGH
rloop_s	See [ai]	Input	Loopback value for a read response
rpoison_s	See [aj]	Input	Indicates that the read data in this transfer has been corrupted

[ac] TBUCFG_RUSER_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ad] TBUCFG_BUSER_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ae] TBUCFG_LOOP_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[af] TBUCFG_LOOP_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ag] [(TBUCFG_DATA_WIDTH/64)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ah] TBUCFG_LOOP_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ai] TBUCFG_LOOP_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[aj] (TBUCFG_DATA_WIDTH / 64)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.2.3 TBU TBM interface signals

The TBU TBM interface signals are based on the AMBA ACE5-Lite signals.

The following table shows the TBU TBM interface signals.

Table A-17 TBU TBM interface signals

Signal	Width	Direction	Description
aclk	1-bit	Input	Clock input
araddr_m	52-bit	Output	Read address
arburst_m	2-bit	Output	Burst type
arcache_m	4-bit	Output	Memory type
ardomain_m	2-bit	Output	Shareability domain
aresetn	1-bit	Input	Active-LOW reset signal
arid_m	See [ak]	Output	Read address ID
arlen_m	8-bit	Output	Burst length
arlock_m	1-bit	Output	Lock type
arprot_m	3-bit	Output	Protection type
arqos_m	4-bit	Output	*Quality of Service* (QoS)
arready_m	1-bit	Input	Read address ready
arregion_m	4-bit	Output	Region identifier
arsize_m	3-bit	Output	Burst size
armmusid_m	See [al]	Output	These signals indicate the StreamID of the originating transaction
armmusecsid_m	1-bit	Output	
arvalid_m	1-bit	Output	Read address valid
awaddr_m	52-bit	Output	Write address
awatop_m	6-bit	Output	Atomic operation
awburst_m	2-bit	Output	Burst type
awcache_m	4-bit	Output	Memory type
awdomain_m	2-bit	Output	Shareability domain
awid_m	See [am]	Output	Write address ID

[ak] `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[al] `TBUCFG_SID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
[am] `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-17 TBU TBM interface signals (continued)

Signal	Width	Direction	Description
awlen_m	8-bit	Output	Burst length
awlock_m	1-bit	Output	Lock type
awprot_m	3-bit	Output	Protection type
awqos_m	4-bit	Output	QoS
awready_m	1-bit	Input	Write address ready
awregion_m	4-bit	Output	Region identifier
awsize_m	3-bit	Output	Burst size
awmmusid_m	See [an]	Output	These signals indicate the StreamID of the originating transaction. The *Generic Interrupt Controller* (GIC) uses these signals to determine the DeviceID of MSIs that originate from upstream masters.
awmmusecsid_m	1-bit	Output	
awvalid_m	1-bit	Output	Write address valid
bid_m	See [ao]	Input	Response ID
bready_m	1-bit	Output	Response ready
bresp_m	2-bit	Input	Write response
bvalid_m	1-bit	Input	Write response valid
rdata_m	See [ap]	Input	Read data
rid_m	See [aq]	Input	Read ID
rlast_m	1-bit	Input	Read last
rready_m	1-bit	Output	Read ready
rresp_m	2-bit	Input	Read response
rvalid_m	1-bit	Input	Read valid
wdata_m	See [ar]	Output	Write data
wlast_m	1-bit	Output	Write last
wready_m	1-bit	Input	Write ready
wstrb_m	See [as]	Output	Write strobes

[an] `TBUCFG_SID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
[ao] `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ap] `TBUCFG_DATA_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[aq] `TBUCFG_ID_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ar] `TBUCFG_DATA_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[as] (`TBUCFG_DATA_WIDTH` / 8)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-17 TBU TBM interface signals (continued)

Signal	Width	Direction	Description
wvalid_m	1-bit	Output	Write valid
aruser_m	See [at]	Output	Read address (AR) channel User signal
awuser_m	See [au]	Output	Write address (AW) channel User signal
wuser_m	See [av]	Output	Write data (W) channel User signal
ruser_m	See [aw]	Input	Read data (R) channel User signal
buser_m	See [ax]	Input	Write response (B) channel User signal
awakeup_m	1-bit	Output	Wakeup signal
arsnoop_m	4-bit	Output	Transaction type of read transaction
awsnoop_m[3:0]	4-bit	Output	Transaction type of write transaction
Read data chunking enable	-	Output	The AXI5 Cache_Stash_Transactions extension defines these signals. See *AMBA® AXI and ACE Protocol Specification*. If TBUCFG_STASH = 0, these signals are ignored.
Read address channel unique ID indicator, active-HIGH			
Loopback value for a read transaction. Reflected back on RLOOP			
Indicates the SMMU flow for managing translation faults			
archunken_m	1-bit	Output	Read data chunking enable
aridunq_m	1-bit	Output	Read address channel unique ID indicator, active-HIGH
arloop_m	See [ay]	Output	Loopback value for a read transaction. Reflected back on **RLOOP**
armmuflow_s	2-bit	Input (Output)	Indicates the SMMU flow for managing translation faults
armpam_m	11-bit	Output	Read address channel MPAM information
awidunq_m	1-bit	Output	Write address channel unique ID indicator, active-HIGH
awloop_m	See [az]	Output	Loopback value for a write transaction
awmmuflow_s	2-bit	Input (Output)	Indicates the SMMU flow for managing translation faults
awmpam_m	11-bit	Output	Write address channel MPAM information
wpoison_m	See [ba]	Output	Indicates that the write data in this transfer has been corrupted

[at] (TBUCFG_ARUSER_WIDTH-+ 4)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[au] (TBUCFG_AWUSER_WIDTH-+ 4)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[av] TBUCFG_WUSER_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[aw] TBUCFG_RUSER_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ax] TBUCFG_BUSER_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ay] If TBUCFG_OT_TRACKER_TYPE is 1, the width of **arloop_m** is (TBUCFG_LOOP_WIDTH + 2)-bit. Otherise, the width of **arloop_m** is TBUCFG_LOOP_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[az] If TBUCFG_OT_TRACKER_TYPE is 1, the width of **awloop_m** is (TBUCFG_LOOP_WIDTH + 2)-bit. Otherwise, the width of **awloop_m** is TBUCFG_LOOP_WIDTH-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.
[ba] (TBUCFG_DATA_WIDTH / 64)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-17 TBU TBM interface signals (continued)

Signal	Width	Direction	Description
bidunq_m	1-bit	Input (Output)	Write response channel unique ID indicator, active-HIGH
bloop_m	See [bb]	Output	Loopback value for a write response
rchunknum_m	See [bc]	Output	Read data chunk number
rchunkstrb_m	See [bd]	Output	Read data chunk strobe
rchunkv_m	1-bit	Output	Valid signal of **RCHUNKNUM** and **RCHUNKSTRB**
ridunq_m	1-bit	Output	Read data channel unique ID indicator, active-HIGH
rloop_m	See [be]	Output	Loopback value for a read response
rpoison_m	See [bf]	Output	Indicates that the read data in this transfer has been corrupted

A.2.4 TBU PMU snapshot interface signals

The following table shows the TBU PMU snapshot interface signals.

Table A-18 TBU PMU snapshot interface signals

Signal	Width	Direction	Description
pmusnapshot_req	1-bit	Input	PMU snapshot request. The PMU snapshot occurs on the rising edge of **pmusnapshot_req**. ——— **Note** ——— Connect to the debug infrastructure of your SoC. ———
pmusnapshot_ack	1-bit	Output	PMU snapshot acknowledge. The TBU uses this signal to acknowledge that the PMU snapshot has occurred. This signal is LOW after reset. ——— **Note** ——— Connect to the debug infrastructure of your SoC. ———

A.2.5 TBU LPI_PD interface signals

The following table shows the TBU LPI_PD interface signals.

[bb] If `TBUCFG_OT_TRACKER_TYPE` is 1, the width of **awloop_m** is (`TBUCFG_LOOP_WIDTH` + 2)-bit. Otherwise, the width of **awloop_m** is `TBUCFG_LOOP_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

[bc] If `TBUCFG_DATA_WIDTH` is 128, then the width of **rchunknum_m** is 8-bit.

If `TBUCFG_DATA_WIDTH` is 256, then the width of **rchunknum_m** is 7-bit.

If `TBUCFG_DATA_WIDTH` is 512, then the width of **rchunknum_m** is 6-bit.

If `TBUCFG_DATA_WIDTH` is 1024, then the width of **rchunknum_m** is 5-bit.

Otherwise, the width of **rchunknum_m** is 1-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

[bd] If `TBUCFG_DATA_WIDTH` is 64, then the width of **rchunkstrb_m** is 1-bit. Otherwise, the width of **rchunkstrb_m** is (`TBUCFG_DATA_WIDTH` / 128)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

[be] If `TBUCFG_OT_TRACKER_TYPE` is 1, the width of **rloop_m** is (`TBUCFG_LOOP_WIDTH` + 2)-bit. Otherwise, the width of **rloop_m** is `TBUCFG_LOOP_WIDTH`-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

[bf] (`TBUCFG_DATA_WIDTH` / 64)-bit. See *2.5.4 ACE-Lite TBU I/O configuration options* on page 2-78.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-19 TBU LPI_PD interface signals

Signal	Width	Direction	Description
qactive_pd	1-bit	Output	Component active
qreqn_pd	1-bit	Input	Quiescence request
qacceptn_pd	1-bit	Output	Quiescence accept
qdeny_pd	1-bit	Output	Quiescence deny

For more information about these signals, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

A.2.6 TBU LPI_CG interface signals

The following table shows the TBU LPI_CG interface signals.

Table A-20 TBU LPI_CG interface signals

Signal	Width	Direction	Description
qactive_cg	1-bit	Output	Component active
qreqn_cg	1-bit	Input	Quiescence request
qacceptn_cg	1-bit	Output	Quiescence accept
qdeny_cg	1-bit	Output	Quiescence deny

For more information about these signals, see the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

A.2.7 TBU DTI interface signals

The following table shows the TBU DTI interface signals.

Table A-21 TBU DTI interface signals

Signal	Width	Direction	Description
tvalid_dti_dn	1-bit	(Master to slave), Output	Flow control signal
tready_dti_dn	1-bit	(Slave to master), Input	Flow control signal
tdata_dti_dn	160-bit	(Master to slave), Output	Message data signal
tlast_dti_dn	1-bit	(Master to slave), Output	Indicates the last cycle of a message
tkeep_dti_dn	20-bit	(Master to slave), Output	Indicates valid bytes
tvalid_dti_up	1-bit	(Slave to master), Input	Flow control signal
tready_dti_up	1-bit	(Master to slave), Output	Flow control signal
tdata_dti_up	160-bit	(Slave to master), Input	Message data signal

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-21 TBU DTI interface signals (continued)

Signal	Width	Direction	Description
tlast_dti_up	1-bit	(Slave to master), Input	Indicates the last cycle of a message
tkeep_dti_up	20-bit	(Slave to master), Input	Indicates valid bytes
twakeup_dti_up	1-bit	(Slave to master), Input	Wakeup signal
twakeup_dti_dn	1-bit	(Master to slave), Output	Wakeup signal

For more information about the DTI signals, see the *AMBA® 4 AXI4-Stream Protocol Specification*.

For more information about DTI protocol messages, see the *AMBA® DTI Protocol Specification*.

A.2.8 TBU interrupt signals

The TBU interrupt signals are edge-triggered. The interrupt controller must detect the rising edge of these signals.

The MMU-700 TBU cannot output these interrupts as *Message Signaled Interrupts* (MSIs). These signals must be connected to an interrupt controller.

The following table shows the TBU interrupt signals.

Table A-22 TBU interrupt signals

Signal	Width	Direction	Description
ras_fhi	1-bit	Output	Fault handling RAS interrupt for a contained error
ras_eri	1-bit	Output	Error recovery RAS interrupt for an uncontained error
ras_cri	1-bit	Output	Critical error interrupt, for an uncontainable uncorrected error
pmu_irpt	1-bit	Output	PMU interrupt

A.2.9 TBU tie-off signals

The TBU tie-off signals are sampled between exiting reset and the LPI_PD interface first entering the Q_RUN state. Ensure that the value of these signals does not change when the LPI_PD interface is in the Q_STOPPED or Q_EXIT state for the first time after exiting reset.

The following table shows the TBU tie-off signals.

Table A-23 TBU tie-off signals

Signal	Width	Direction	Description
ns_sid_high	See [bg]	Input	Provides the high-order StreamID bits for all transactions with a Non-secure StreamID that pass through the TBU
s_sid_high	See [bh]	Input	Provides the high-order StreamID bits for all transactions with a Secure StreamID that pass through the TBU

bg (24 - `TBUCFG_SID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.
bh (24 - `TBUCFG_SID_WIDTH`-bit. See *2.5.7 Common LTI TBU and ACE-Lite TBU configuration options* on page 2-80.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-23 TBU tie-off signals (continued)

Signal	Width	Direction	Description
max_tok_trans	See [bi]	Input	Indicates the number of DTI translation tokens to request when connecting to the TCU, minus 1
pcie_mode	1-bit	Input	You must tie this signal HIGH when the TBU is connected to a PCIe interface. When this signal is HIGH, the TBU interprets the input AXI memory types as encoding PCI 'No Snoop' information. In order for the TBU to provide correct operation, transactions from the PCIe interface must be delivered to the TBU with the following AXI memory types: **Normal Non-Cacheable Bufferable** When 'No Snoop' is set for the transaction **Write-Back** When 'No Snoop' is not set for the transaction This TBU behavior is a requirement of the *Arm® Server Base System Architecture 7.0 Platform Design Document*. If this signal is HIGH, the attributes of TBS interface transactions are always combined with the translation attributes, even if stage 1 translation is enabled. That is, the transaction attributes are always calculated as if the DTI_TBU_TRANS_RESP.STRW field is EL1-S2, regardless of the actual STRW value. If this signal is HIGH, the input attribute and shareability override information in the ATTR_OVR field of the DTI_TBU_TRANS_RESP message is ignored. For SMMUv3, PCIe masters do not support this feature.
sec_override	1-bit	Input	When HIGH, certain registers are accessible to Non-secure accesses from reset, as the TBU_SCR register settings describe. See *3.14.2 TBU_SCR register* on page 3-145.
ecorevnum[3:0]	4-bit	Input	Tie this signal to 0 unless directed otherwise by Arm
utlb_roundrobin	1-bit	Input	Defines the Micro TLB entry replacement policy. When LOW, the Micro TLB uses a *Pseudo Least Recently Used* (PLRU) replacement policy. This policy typically provides the best average performance. When HIGH, the Micro TLB uses a round-robin replacement policy. With this policy, the oldest entry is evicted when the Micro TLB is full. Tie this signal HIGH if you want to prevent newer translations from being evicted, even if older translations have been used more recently. Otherwise, tie this signal LOW.
poison_support	1-bit	Input	When LOW, the ACE-Lite TBU does not drive the **wpoison** signal HIGH in any condition. It handles the RAS errors in the write data buffer by generating a RAS error interrupt and a *Service Failure Mode* (SFM) request to the TCU. When HIGH, **poison_support** drives **wpoison** when driving the write data that has a RAS error.

A.2.10 TBU ELA debug signals

The MMU-700 TBU includes *Embedded Logic Analyzer* (ELA) debug signals.

The following table shows the ELA enable signal.

[bi] ($\log_2$`TBUCFG_XLATE_SLOTS`)-bit. See *2.5.8 TBU buffer configuration options* on page 2-81.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-24 ELA enable signal

Signal	Width	Direction	Description
ela_enable	1-bit	Input	**ela_enable** is an asynchronous input port. When TBUCFG_USE_ELA_DEBUG is 0, the SMMU ignores the value of the signal. When TBUCFG_USE_ELA_DEBUG is 1, **ela_enable** acts as a clock enable for the TBU ELA observation interface. If ELA debug is required, drive **ela_enable** HIGH. If ELA debug is not required, drive **ela_enable** LOW to reduce the dynamic power consumption of the SMMU.

ACE-Lite TBU ELA debug signals

The following table shows the ACE-Lite TBU ELA debug signals.

Table A-25 ACE-Lite TBU ELA debug signals

Signal	Width	Direction	Description
Signal group 0 signals			
signalgrp0	128-bit	Output	See *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209.
sigqual0	4-bit		
sigclken0	1-bit		
Signal group 1 signals			
signalgrp1	128-bit	Output	See *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209.
sigqual1	4-bit		
sigclken1	1-bit		
Signal group 2 signals			
signalgrp2	128-bit	Output	See *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209.
sigqual2	4-bit		
sigclken2	1-bit		
Signal group 3 signals			
signalgrp3	128-bit	Output	See *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209.
sigqual3	4-bit		
sigclken3	1-bit		
Signal group 4 signals			
signalgrp4	128-bit	Output	See *B.2 ACE-Lite TBU observation interfaces* on page Appx-B-209.
sigqual4	4-bit		
sigclken4	1-bit		

LTI TBU ELA debug signals

The following table shows the LTI TBU ELA debug signals.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-26 LTI TBU ELA debug signals

Signal	Width	Direction	Description
Signal group 0 signals			
signalgrp0	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual0	4-bit		
sigclken0	1-bit		
Signal group 1 signals			
signalgrp1	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual1	4-bit		
sigclken1	1-bit		
Signal group 2 signals			
signalgrp2	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual2	4-bit		
sigclken2	1-bit		
Signal group 3 signals			
signalgrp3	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual3	4-bit		
sigclken3	1-bit		
Signal group 4 signals			
signalgrp4	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual4	4-bit		
sigclken4	1-bit		
Signal group 5 signals			
signalgrp5	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual5	4-bit		
sigclken5	1-bit		
Signal group 6 signals			
signalgrp6	128-bit	Output	See *B.3 LTI TBU observation interfaces* on page Appx-B-213.
sigqual6	4-bit		
sigclken6	1-bit		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.3 TCU and TBU shared signals

This section describes the MMU-700 shared TCU and TBU signals.

This section contains the following subsection:

A.3.1 TCU and TBU test and debug signals

The test and debug signals are common to the TCU and TBU.

The following table shows the test and debug signals.

Table A-27 Test and debug signals

Signal	Width	Direction	Description
dftcgen	1-bit	Input	Clock gate enable. To enable architectural clock gates for the **aclk** clock, set this signal HIGH during scan shift.
dftrstdisable	1-bit	Input	Reset disable. To disable reset, set this signal HIGH during scan shift.
dftramhold	1-bit	Input	Preserve RAM state. To preserve the state of the RAMs and their connected registers, set this signal HIGH during scan shift.
MBISTRESETN	1-bit	Input	MBIST mode reset. This active-LOW signal is encoded as follows: **0** Reset MBIST functional logic. **1** Normal operation.
MBISTREQ	1-bit	Input	MBIST test request. This signal is encoded as follows: **0** Normal operation. **1** Enable MBIST testing.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.4 DTI signals

This section describes the MMU-700 DTI signals.

This section contains the following subsections:

A.4.1 DTI interconnect switch signals

The DTI interconnect switch provides signals for each of its interfaces.

The switch provides one DN_S*n* slave downstream interface per slave interface. The following table shows the DN_S*n* signals.

Table A-28 DTI interconnect switch DN_S*n* interface signals

Signal	Width	Direction	Description
tvalid_dti_dn_s*n*	1-bit	Input, slave to master	Flow control signal
tready_dti_dn_s*n*	1-bit	Output, master to slave	Flow control signal
tdata_dti_dn_s*n*	See [bj]	Slave to master	Message data signal
tid_dti_dn_s*n*	See [bk]	Input, slave to master	Indicates the master that initiated the message
tlast_dti_dn_s*n*	1-bit	Input, slave to master	Indicates the last cycle of a message
tkeep_dti_dn_s*n*	See [bl]	Input, slave to master	Indicates valid bytes
twakeup_dti_dn_s*n*	1-bit	Input, slave to master	Wakeup signal

The switch provides one UP_S*n* slave upstream interface per slave interface. The following table shows the UP_S*n* signals.

Table A-29 DTI interconnect switch UP_S*n* interface signals

Signal	Width	Direction	Description
tvalid_dti_up_s*n*	1-bit	Output, master to slave	Flow control signal
tready_dti_up_s*n*	1-bit	Input, slave to master	Flow control signal
tdata_dti_up_s*n*	See [bm]	Output, master to slave	Message data signal
tdest_dti_up_s*n*	See [bn]	Output, master to slave	Indicates the master that initiated the message
tlast_dti_up_s*n*	1-bit	Output, master to slave	Indicates the last cycle of a message

[bj] `DATA_WIDTH` (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.
[bk] `ID_WIDTH` = ($\log_2$(total number of masters being switched))-bit.
[bl] (`DATA_WIDTH` / 8)-bit.
[bm] `DATA_WIDTH` (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.
[bn] `ID_WIDTH` = ($\log_2$(total number of masters being switched))-bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-29 DTI interconnect switch UP_S*n* interface signals (continued)

Signal	Width	Direction	Description
tkeep_dti_up_s*n*	See [bo]	Output, master to slave	Indicates valid bytes
twakeup_dti_up_s*n*	1-bit	Output, master to slave	Wakeup signal

The switch provides a DN_M master downstream interface. The following table shows the DN_M signals.

Table A-30 DTI interconnect switch DN_M interface signals

Signal	Width	Direction	Description
tvalid_dti_dn_m	1-bit	Output, slave to master	Flow control signal
tready_dti_dn_m	1-bit	Input, master to slave	Flow control signal
tdata_dti_dn_m	See [bp]	Output, slave to master	Message data signal
tid_dti_dn_m	See [bq]	Output, slave to master	Indicates the master that initiated the message
tlast_dti_dn_m	1-bit	Output, slave to master	Indicates the last cycle of a message
tkeep_dti_dn_m	See [br]	Output, slave to master	Indicates valid bytes
twakeup_dti_dn_m	1-bit	Output, slave to master	Wakeup signal

The switch provides an UP_M master upstream interface. The following table shows the UP_M signals.

Table A-31 DTI interconnect switch UP_M interface signals

Signal	Width	Direction	Description
tvalid_dti_up_m	1-bit	Input, master to slave	Flow control signal
tready_dti_up_m	1-bit	Output, slave to master	Flow control signal
tdata_dti_up_m	See [bs]	Input, master to slave	Message data signal
tdest_dti_up_m	See [bt]	Input, master to slave	Indicates the master that initiated the message
tlast_dti_up_m	1-bit	Input, master to slave	Indicates the last cycle of a message
tkeep_dti_up_m	See [bu]	Input, master to slave	Indicates valid bytes
twakeup_dti_up_m	1-bit	Input, slave to master	Wakeup signal

[bo] (`DATA_WIDTH` / 8)-bit.
[bp] `DATA_WIDTH` (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.
[bq] `ID_WIDTH` = ($\log_2$(total number of masters being switched))-bit.
[br] (`DATA_WIDTH` / 8)-bit.
[bs] `DATA_WIDTH` (width of the payload). Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.
[bt] `ID_WIDTH` = ($\log_2$(total number of masters being switched))-bit.
[bu] (`DATA_WIDTH` / 8)-bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.4.2 DTI interconnect sizer signals

The DTI interconnect sizer provides signals for each of its interfaces.

The sizer provides an LPI_CG clock gating interface. The following table shows the LPI_CG signals.

Table A-32 DTI interconnect sizer LPI_CG interface signals

Signal	Width	Direction	Description
qactive_cg	1-bit	Output	Component active
qreqn_cg	1-bit	Input	Quiescence request
qacceptn_cg	1-bit	Output	Quiescence accept
qdeny_cg	1-bit	Output	Quiescence deny

The sizer provides a DN_S slave downstream interface. The following table shows the DN_S signals.

Table A-33 DTI interconnect sizer DN_S interface signals

Signal	Width	Direction	Description
tvalid_dti_dn_s	1-bit	Input	Flow control signal
tready_dti_dn_s	1-bit	Output	Flow control signal
tdata_dti_dn_s	See [bv]	Input	Message data signal
tid_dti_dn_s	See [bw]	Input	Indicates the master that initiated the message
tlast_dti_dn_s	1-bit	Input	Indicates the last cycle of a message
tkeep_dti_dn_s	See [bx]	Input	Indicates valid bytes
twakeup_dti_dn_s	1-bit	Input	Wakeup signal

The sizer provides an UP_S slave upstream interface. The following table shows the UP_S signals.

Table A-34 DTI interconnect sizer UP_S interface signals

Signal	Width	Direction	Description
tvalid_dti_up_s	1-bit	Output	Flow control signal
tready_dti_up_s	1-bit	Input	Flow control signal
tdata_dti_up_s	See [by]	Output	Message data signal
tdest_dti_up_s	See [bz]	Output	Indicates the master that initiated the message

[bv] `INPUT_DATA_WIDTH`. Can be 160-bit, 80-bit, 32-bit, or 8-bit, depending on the sizing of the payload before it.
[bw] `ID_WIDTH` = ($\log_2$(total number of masters that are connected to the sizer))-bit.
[bx] (`INPUT_DATA_WIDTH` / 8)-bit.
[by] `INPUT_DATA_WIDTH`. Can be 160-bit, 80-bit, 32-bit, or 8-bit.
[bz] `ID_WIDTH` = ($\log_2$(total number of masters that are connected to the sizer))-bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-34 DTI interconnect sizer UP_S interface signals (continued)

Signal	Width	Direction	Description
tlast_dti_up_s	1-bit	Output	Indicates the last cycle of a message
tkeep_dti_up_s	See [ca]	Output	Indicates valid bytes
twakeup_dti_up_s	1-bit	Output	Wakeup signal

The sizer provides a DN_M master downstream interface. The following table shows the DN_M signals.

Table A-35 DTI interconnect sizer DN_M interface signals

Signal	Width	Direction	Description
tvalid_dti_dn_m	1-bit	Input	Flow control signal
tready_dti_dn_m	1-bit	Input	Flow control signal
tdata_dti_dn_m	See [cb]	Output	Message data signal
tid_dti_dn_m	See [cc]	Output	Indicates the master that initiated the message
tlast_dti_dn_m	1-bit	Output	Indicates the last cycle of a message
tkeep_dti_dn_m	See [cd]	Output	Indicates valid bytes
twakeup_dti_dn_m	1-bit	Input	Wakeup signal

The sizer provides an UP_M master upstream interface. The following table shows the UP_M signals.

Table A-36 DTI interconnect sizer UP_M interface signals

Signal	Width	Direction	Description
tvalid_dti_up_m	1-bit	Input	Flow control signal
tready_dti_up_m	1-bit	Output	Flow control signal
tdata_dti_up_m	See [ce]	Input	Message data signal
tdest_dti_up_m	See [cf]	Input	Indicates the master that initiated the message
tlast_dti_up_m	1-bit	Input	Indicates the last cycle of a message
tkeep_dti_up_m	See [cg]	Input	Indicates valid bytes
twakeup_dti_up_m	1-bit	Input	Wakeup signal

[ca] `(INPUT_DATA_WIDTH / 8)`-bit
[cb] `OUTPUT_DATA_WIDTH`-bit. `OUTPUT_DATA_WIDTH` can be 160-bit, 80-bit, 32-bit, or 8-bit.
[cc] `ID_WIDTH` = ($\log_2$(total number of masters that are connected to the sizer))-bit.
[cd] `(OUTPUT_DATA_WIDTH / 8)`-bit.
[ce] `OUTPUT_DATA_WIDTH`. Can be 160-bit, 80-bit, 32-bit, or 8-bit.
[cf] `ID_WIDTH` = ($\log_2$(total number of masters that are connected to the sizer))-bit.
[cg] `(OUTPUT_DATA_WIDTH / 8)`-bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

A.4.3 DTI interconnect register slice signals

The DTI interconnect register slice provides signals for each of its interfaces.

The register slice provides an LPI_CG clock gating interface. The following table shows the LPI_CG signals.

Table A-37 DTI interconnect register slice LPI_CG interface signals

Signal	Width	Direction	Description
qactive_cg	1-bit	Output	Component active
qreqn_cg	1-bit	Input	Quiescence request
qacceptn_cg	1-bit	Output	Quiescence accept
qdeny_cg	1-bit	Output	Quiescence deny

The register slice provides a DN_S slave downstream interface. The following table shows the DN_S signals.

Table A-38 DTI interconnect register slice DN_S interface signals

Signal	Width	Direction	Description
tvalid_dti_dn_s	1-bit	Slave to master	Flow control signal
tready_dti_dn_s	1-bit	Master to slave	Flow control signal
tdata_dti_dn_s	See [ch]	Slave to master	Message data signal
tid_dti_dn_s	See [ci]	Slave to master	Indicates the master that initiated the message
tlast_dti_dn_s	1-bit	Slave to master	Indicates the last cycle of a message
tkeep_dti_dn_s	See [cj]	Slave to master	Indicates valid bytes

The register slice provides an UP_S slave upstream interface. The following table shows the UP_S signals.

Table A-39 DTI interconnect register slice UP_S interface signals

Signal	Width	Direction	Description
tvalid_dti_up_s	1-bit	Master to slave	Flow control signal
tready_dti_up_s	1-bit	Slave to master	Flow control signal
tdata_dti_up_s	See [ck]	Master to slave	Message data signal
tdest_dti_up_s	See [cl]	Master to slave	Indicates the master that initiated the message

[ch] `DATA_WIDTH` of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.
[ci] `ID_WIDTH` = ($\log_2$(total number of masters that are connected to the register slice))-bit.
[cj] (`DATA_WIDTH` / 8)-bit.
[ck] `DATA_WIDTH` of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.
[cl] `ID_WIDTH` = ($\log_2$(total number of masters that are connected to the register slice))-bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table A-39 DTI interconnect register slice UP_S interface signals (continued)

Signal	Width	Direction	Description
tlast_dti_up_s	1-bit	Master to slave	Indicates the last cycle of a message
tkeep_dti_up_s	See [cm]	Master to slave	Indicates valid bytes

The register slice provides a DN_M master downstream interface. The following table shows the DN_M signals.

Table A-40 DTI interconnect register slice DN_M interface signals

Signal	Width	Direction	Description
tvalid_dti_dn_m	1-bit	Slave to master	Flow control signal
tready_dti_dn_m	1-bit	Master to slave	Flow control signal
tdata_dti_dn_m	See [cn]	Slave to master	Message data signal
tid_dti_dn_m	See [co]	Slave to master	Indicates the master that initiated the message
tlast_dti_dn_m	1-bit	Slave to master	Indicates the last cycle of a message
tkeep_dti_dn_m	See [cp]	Slave to master	Indicates valid bytes

The register slice provides an UP_M master upstream interface. The following table shows the UP_M signals.

Table A-41 DTI interconnect register slice UP_M interface signals

Signal	Width	Direction	Description
tvalid_dti_up_m	1-bit	Master to slave	Flow control signal
tready_dti_up_m	1-bit	Slave to master	Flow control signal
tdata_dti_up_m	See [cq]	Master to slave	Message data signal
tdest_dti_up_m	See [cr]	Master to slave	Indicates the master that initiated the message
tlast_dti_up_m	1-bit	Master to slave	Indicates the last cycle of a message
tkeep_dti_up_m	See [cs]	Master to slave	Indicates valid bytes

[cm] (DATA_WIDTH / 8)-bit
[cn] DATA_WIDTH of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.
[co] ID_WIDTH = ($\log_2$(total number of masters that are connected to the register slice))-bit.
[cp] (DATA_WIDTH / 8)-bit.
[cq] DATA_WIDTH of the register slice. Can be 160-bit, 80-bit, 32-bit, or 8-bit.
[cr] ID_WIDTH = ($\log_2$(total number of masters that are connected to the register slice))-bit.
[cs] (DATA_WIDTH / 8)-bit.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Appendix B
ELA signal descriptions

This section describes the **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals of the TCU and TBU components that are used to interface with external ELA.

It contains the following sections:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

B.1 TCU observation interfaces

This section describes the TCU observation interfaces, **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals that are used to interface to an external CoreSight ELA-600 Embedded Logic Analyzer. **<n>** represents the number in the signal name.

Signal group output ports are present on each component. However, only a subset is used.

The **SIGCLKEN<n>** signal is set to 1 for the signal groups in the 'Enabled signal groups' column in the following table. Groups that are not enabled have their **SIGCLKEN<n>** signals set to 0. If **ela_enable** is driven LOW, all **SIGCLKEN<n>** signals are set to 0.

The following table shows the signal group output ports that are valid for the TCU.

Table B-1 Number of signal groups per module for the TCU

Component	Parameter	Enabled signal groups	Total
TCU	TCUCFG_QTW_DATA_WIDTH <= 128	0, 1, 2, 3, 4, 5, 6, 10	8
	TCUCFG_QTW_DATA_WIDTH == 256	0, 1, 2, 3, 4, 5, 6, 7, 10, 11	10
	TCUCFG_QTW_DATA_WIDTH == 512	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	12

The following table shows the **SIGNALGPR<n>** bits for the signal groups of the TCU.

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU-700 delays sections of the signal by a cycle so that the ELA can sample 128-bit chunks of the data one cycle after another. The *Number of cycles of delay* column in the table indicates the number of cycles, from when the signal is observable on a MMU-700 interface, to when the signal is observable on the ELA observation interface.

Table B-2 TCU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
0	[127:0]	**tdata_dti_dn[127:0]**	1'b0, (**tready_dti_dn** AND **tvalid_dti_dn**), **tready_dti_dn**, **tvalid_dti_dn**	1
1	[127:0]	**tdata_dti_up[127:0]**	1'b0, (**tready_dti_up** AND **tvalid_dti_up**), **tready_dti_up**, **tvalid_dti_up**	1

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-2 TCU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
2	[127:124]	Unused	-	-
	[123:118]	**tid_dti_dn**	**tvalid_dti_up**, (**tready_dti_up** AND **tvalid_dti_up**), **tvalid_dti_dn**, (**tready_dti_dn** AND **tvalid_dti_dn**)	0
	[117:86]	**tdata_dti_dn[159:128]**		
	[85:66]	**tkeep_dti_dn**		
	[65]	**tlast_dti_dn**		
	[64]	**twakeup_dti_dn**		
	[63]	**tready_dti_dn**		
	[62]	**tvalid_dti_dn**		
	[61:56]	**tdest_dti_up**		
	[55:24]	**tdata_dti_up[159:128]**		
	[23:4]	**tkeep_dti_up**		
	[3]	**tlast_dti_up**		
	[2]	**twakeup_dti_up**		
	[1]	**tready_dti_up**		
	[0]	**tvalid_dti_up**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-2 TCU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
3	[127]	Unused	-	-
	[126]	**ridunq_qtw**	**rvalid_qtw**, (**rready_qtw** AND **rvalid_qtw**), **arvalid_qtw**, (**arready_qtw** AND **arvalid_qtw**)	1
	[125:118]	**rpoison_qtw**		
	[117]	**rlast_qtw**		
	[116:106]	**rid_qtw**		
	[105]	**rready_qtw**		
	[104]	**rvalid_qtw**		
	[103:93]	**arid_qtw**		
	[92:92]	**aridunq_qtw**		
	[91:81]	**armpam_qtw**		
	[80:79]	**ardomain_qtw**		
	[78:75]	**aruser_qtw**		
	[74:71]	**arqos_qtw**		
	[70:67]	**arcache_qtw**		
	[66:65]	**arburst_qtw**		
	[64:62]	**arsize_qtw**		
	[61:54]	**arlen_qtw**		
	[53:2]	**araddr_qtw**		
	[1]	**arready_qtw**		
	[0]	**arvalid_qtw**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-2 TCU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
4	[127]	Unused	-	-
	[126]	**crready_qtw**	`1'b0`, (**crready_qtw** AND **crvalid_qtw**), (**bready_qtw** AND **bvalid_qtw**), (**awready_qtw** AND **awvalid_qtw**)	1
	[125]	**crvalid_qtw**		
	[124:114]	**bid_qtw**		
	[113]	**bidunq_qtw**		
	[112]	**bready_qtw**		
	[111]	**bvalid_qtw**		
	[110]	**awakeup_qtw**		
	[109:99]	**awid_qtw**		
	[98:93]	**awatop_qtw**		
	[92]	**awidunq_qtw**		
	[91:81]	**awmpam_qtw**		
	[80:79]	**awdomain_qtw**		
	[78:75]	**awuser_qtw**		
	[74:71]	**awqos_qtw**		
	[70:67]	**awcache_qtw**		
	[66:65]	**awburst_qtw**		
	[64:62]	**awsize_qtw**		
	[61:54]	**awlen_qtw**		
	[53:2]	**awaddr_qtw**		
	[1]	**awready_qtw**		
	[0]	**awvalid_qtw**		
5	[127]	**syscoack_qtw**	`2'b00`, (**wready_qtw** AND **wvalid_qtw**), (**acready_qtw** AND **acvalid_qtw**)	1
	[126]	**syscoreq_qtw**		
	[125:62]	**wstrb_qtw**		
	[61]	**wlast_qtw**		
	[60]	**wready_qtw**		
	[59]	**wvalid_qtw**		
	[58]	**acwakeup_qtw**		
	[57:6]	**acaddr_qtw**		
	[5:2]	**acvmidext_qtw**		
	[1]	**acready_qtw**		
	[0]	**acvalid_qtw**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-2 TCU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
6	[127:0]	**rdata_qtw[127:0]**	`1'b0`, (**rready_qtw** AND **rvalid_qtw**), **rready_qtw**, **rvalid_qtw**	1
7	[127:0]	**rdata_qtw[255:128]**	`1'b0`, (**rready_qtw** AND **rvalid_qtw**), **rready_qtw**, **rvalid_qtw**	2
8	[127:0]	**rdata_qtw[383:256]**	`1'b0`, (**rready_qtw** AND **rvalid_qtw**), **rready_qtw**, **rvalid_qtw**	3
9	[127:0]	**rdata_qtw[511:384]**	`1'b0`, (**rready_qtw** AND **rvalid_qtw**), **rready_qtw**, **rvalid_qtw**	4
10	[127:0]	**wdata_qtw_demuxed**[ct]**[127:0]**	`1'b0`, (**wready_qtw** AND **wvalid_qtw**), **wready_qtw**, **wvalid_qtw**	1
11	[127:0]	**wdata_qtw_demuxed[255:128]**	`1'b0`, (**wready_qtw** AND **wvalid_qtw**), **wready_qtw**, **wvalid_qtw**	2

[ct] When the `TCUQTWDATAWIDTH` parameter is set to 512, the **wdate_qtw_demuxed** signal contains the active 256 bits of the 512-bit bus. The **wstrb_qtw** signal remains as 64 bits and is unmodified. See *2.5 Configuration options and methodology* on page 2-75.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

B.2 ACE-Lite TBU observation interfaces

This section describes the ACE-Lite TBU observation interfaces, **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals that are used to interface to an external CoreSight ELA-600 Embedded Logic Analyzer. **<n>** represents the number in the signal name.

Signal group output ports are present on each component. However, only a subset is used.

The **SIGCLKEN<n>** signal is set to 1 for the signal groups in the 'Enabled signal groups' column in the following table. Groups that are not enabled have their **SIGCLKEN<n>** signals set to 0. If **ela_enable** is driven LOW, all **SIGCLKEN<n>** signals are set to 0.

The following table shows the signal group output ports that are valid for the ACE-Lite TBU.

Table B-3 Number of signal groups per module for the ACE-Lite TBU

Component	Parameter	Enabled signal groups	Total
ACE-Lite TBU		0, 1, 2, 3, 4	5

The following table shows the **SIGNALGPR<n>** bits for the signal groups of the ACE-Lite TBU.

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU-700 delays sections of the signal by a cycle so that the ELA can sample 128-bit chunks of the data one cycle after another. The *Number of cycles of delay* column in the table indicates the number of cycles, from when the signal is observable on a MMU-700 interface, to when the signal is observable on the ELA observation interface.

Table B-4 ACE-Lite TBU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
0	[127:123]	**awuser_TLBLOC**	1'b0, (**awready_m** AND **awvalid_m**), **awready_m**, **awvalid_m**	1
	[122]	**awidunq_m**		
	[121:116]	**awatop_m**		
	[115:114]	**awdomain_m**		
	[113:110]	**awqos_m**		
	[109:107]	**awprot_m**		
	[106:103]	**awcache_m**		
	[102:101]	**awburst_m**		
	[100:98]	**awsize_m**		
	[97:90]	**awlen_m**		
	[89:86]	**awregion_m**		
	[85:54]	**awid_m**		
	[53:2]	**awaddr_m**		
	[1]	**awready_m**		
	[0]	**awvalid_m**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-4 ACE-Lite TBU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
1	[127:114]	Unused	-	-
	[113:103]	**awmpam_m**	`1'b0`, (**awready_m** AND **awvalid_m**), **awready_m**, **awvalid_m**	2
	[102:83]	**awmmusid_m**		
	[82]	**awmmusecsid_m**		
	[81:72]	**awloop_m**		
	[71]	**awstashlpiden_m**		
	[70:66]	**awstashlpid_m**		
	[65]	**awstashniden_m**		
	[64:54]	**awstashnid_m**		
	[53:2]	**awaddr_m**		
	[1]	**awready_m**		
	[0]	**awvalid_m**		
2	[127:117]	Unused	-	-
	[116:65]	**araddr_m**	`1'b0`, (**arready_m** AND **arvalid_m**), **arready_m**, **arvalid_m**	1
	[64:63]	**ardomain_m**		
	[62:59]	**arqos_m**		
	[58:56]	**arprot_m**		
	[55:52]	**arcache_m**		
	[51:50]	**arburst_m**		
	[49:47]	**arsize_m**		
	[46:39]	**arlen_m**		
	[38:34]	**aruser_TLBLOC**		
	[33:2]	**arid_m**		
	[1]	**arready_m**		
	[0]	**arvalid_m**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-4 ACE-Lite TBU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
3	[127:113]	Unused	-	-
	[112:]	**archunken_m**	`1'b0`, (**arready_m** AND **arvalid_m**), **arready_m**, **arvalid_m**	2
	[111:101]	**armpam_m**		
	[100:91]	**arloop_m**		
	[90]	**aridunq_m**		
	[89:70]	**armmusid_m**		
	[69]	**armmusecsid_m**		
	[68:65]	**arregion_m**		
	[64:63]	**ardomain_m**		
	[62:59]	**arqos_m**		
	[58:56]	**arprot_m**		
	[55:52]	**arcache_m**		
	[51:50]	**arburst_m**		
	[49:47]	**arsize_m**		
	[46:39]	**arlen_m**		
	[38:34]	**aruser_TLBLOC**		
	[33:2]	**arid_m**		
	[1]	**arready_m**		
	[0]	**arvalid_m**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-4 ACE-Lite TBU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
4	[127:98]	Unused	-	-
	[97]	**wlast_m**	`1'b0`, (**wready_m** AND **wvalid_m**), (**bready_m** AND **bvalid_m**), (**rready_m** AND **rvalid_m**)	1
	[96]	**wready_m**		
	[95]	**wvalid_m**		
	[94:85]	**bloop_m**		
	[84:84]	**bidunq_m**		
	[83:82]	**bresp_m**		
	[81:50]	**bid_m**		
	[49]	**bready_m**		
	[48]	**bvalid_m**		
	[47]	**ridunq_m**		
	[46:37]	**rloop_m**		
	[36]	**rlast_m**		
	[35:34]	**rresp_m**		
	[33:2]	**rid_m**		
	[1]	**rready_m**		
	[0]	**rvalid_m**		
5	[127:0]	Unused	-	-
6				
7				
8				
9				
10				
11				

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

B.3 LTI TBU observation interfaces

This section describes the LTI TBU observation interfaces, **SIGNALGRP<n>**, **SIGQUAL<n>**, and **SIGCLKEN<n>** signals that are used to interface to an external CoreSight ELA-600 Embedded Logic Analyzer. **<n>** represents the number in the signal name.

Signal group output ports are present on each component. However, only a subset is used.

The **SIGCLKEN<n>** signal is set to 1 for the signal groups in the 'Enabled signal groups' column in the following table. Groups that are not enabled have their **SIGCLKEN<n>** signals set to 0. If **ela_enable** is driven LOW, all **SIGCLKEN<n>** signals are set to 0.

The following table shows the signal group output ports that are valid for the LTI TBU.

Table B-5 Number of SignalGroups per module for the LTI TBU

Component	Parameter	Enabled signal groups	Total
LTI TBU	When `TBUCFG_LTI_LOOP_WIDTH` <= 128 bits	`0`, `1`, `2`, `3`, `5`	`5`
	When `TBUCFG_LTI_LOOP_WIDTH` > 128 bit	`0`, `1`, `2`, `3`, `4`, `5`, `6`	`7`

The following table shows the **SIGNALGPR<n>** bits for the signal groups of the LTI TBU.

Some buses, if configured to be larger than the 128-bit signal group width, are spread across multiple groups. The MMU-700 delays sections of the signal by a cycle so that the ELA can sample 128-bit chunks of the data one cycle after another. The *Number of cycles of delay* column in the table indicates the number of cycles, from when the signal is observable on a MMU-700 interface, to when the signal is observable on the ELA observation interface.

Table B-6 LTI TBU observation interface signals

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
0	[127:112]	**latlbloc**	`3'b000`, **lavalid**	0
	[111:80]	**laid**		
	[79:78]	**laflow**		
	[77:74]	**laattr**		
	[73:70]	**latrans**		
	[69:67]	**laprot**		
	[66:65]	**lacredit**		
	[64:1]	**laaddr**		
	[0]	**lavalid**		

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-6 LTI TBU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
1	[127]	Unused	-	-
	[126:122]	**laog**	`3'b000`, **lavalid**	1
	[121]	**laogv**		
	[120:101]	**lassid**		
	[100]	**lassidv**		
	[99:80]	**lasid**		
	[79]	**lasecsid**		
	[78]	**lavc**		
	[77:74]	**laattr**		
	[73:70]	**latrans**		
	[69:67]	**laprot**		
	[66:65]	**lacredit**		
	[64:1]	**laaddr**		
	[0]	**lavalid**		
2	[127:121]	Unused	-	-
	[120]	**lmaskclose**	`2'b00`, **lcvalid**, **lrvalid**	0
	[119]	**lmactive**		
	[118]	**lmopenack**		
	[117]	**lmopenreq**		
	[116]	**lcctag**		
	[115]	**lccredit**		
	[114]	**lcvalid**		
	[113:103]	**lrmpam**		
	[102:99]	**lrhwattr**		
	[98:95]	**lrattr**		
	[94:43]	**lraddr**		
	[42:40]	**lrprot**		
	[39:37]	**lrresp**		
	[36]	**lrctag**		
	[35:4]	**lrid**		
	[3]	**lrvc**		
	[2:1]	**lrcredit**		
	[0]	**lrvalid**		
3	[127:0]	**laloop[127:0]**	`3'b000`, **lavalid**	1

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table B-6 LTI TBU observation interface signals (continued)

SIGNALGRP<n>	Bits	Signal name	SIGQUAL<n> 4'b{MSB..LSB}	Number of cycles of delay
4	[127:0]	**laloop[255:128]**	3'b000, **lavalid**	2
5	[127:0]	**lrloop[127:0]**	3'b000, **lrvalid**	1
6	[127:0]	**lrloop[255:128]**	3'b000, **lrvalid**	2
7	[127:0]	Unused	-	-
8				
9				
10				
11				

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Appendix C
Software initialization examples

This appendix provides examples of how software can initialize and enable the MMU-700.

It contains the following sections:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

C.1 Initializing the SMMU

Software must initialize the MMU-700 before you can use it.

The MMU-700 supports Secure and Non-secure translation worlds. This section defines how to initialize Non-secure translation. The procedures for initializing Secure translation are similar, and require you to access the corresponding MMU-700 Secure registers.

Note

This section does not describe how to create translation tables. For more information, see the *Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile*.

For more information about MMU-700 initialization, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

This section contains the following subsections:

C.1.1 Allocating the Command queue

The MMU-700 uses the Command queue to receive commands. Software must allocate memory for the Command queue and configure the appropriate registers in the SMMU.

To allocate the Command queue, ensure that your software performs the following steps:

Procedure

1. Allocate memory for the Command queue.
2. Configure the Command queue size and base address by writing to the SMMU_CMDQ_BASE register.

   Note

   The queue size can affect how many bits of the SMMU_CMDQ_CONS and SMMU_CMDQ_PROD indices are writeable. It is therefore important that you perform this step before writing to SMMU_CMDQ_CONS and SMMU_CMDQ_PROD.

3. Set the queue read index in SMMU_CMDQ_CONS and the queue write index in SMMU_CMDQ_PROD to 0.

   Note

   Setting the queue read index and the queue write index to the same value indicates that the queue is empty.

C.1.2 Allocating the Event queue

The MMU-700 uses the Event queue to signal events. Software must allocate memory for the Event queue and configure the appropriate registers in the MMU.

To allocate the Event queue, ensure that your software performs the following steps:

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Procedure

1. Allocate memory for the Event queue.
2. Configure the Event queue size and base address by writing to the SMMU_EVENTQ_BASE register.

   Note

   The queue size can affect how many bits of the SMMU_EVENTQ_CONS and SMMU_EVENTQ_PROD indices are writeable. It is therefore important that you perform this step before writing to SMMU_EVENTQ_CONS and SMMU_EVENTQ_PROD.

3. Set the queue read index in SMMU_EVENTQ_CONS and the queue write index in SMMU_EVENTQ_PROD to 0.

   Note

   Setting the queue read index and the queue write index to the same value indicates that the queue is empty.

C.1.3 Configuring the Stream table

The Stream table is a configuration structure in memory that uses a *Context Descriptor* (CD) to locate translation data for a transaction. Software must allocate memory for the Stream table, configure the table format, and populate the table with *Stream Table Entries* (STEs).

To configure the Stream table, ensure that your software performs the following steps:

Procedure

1. Allocate memory for the Stream table.
2. Configure the format and size of the Stream table by writing to SMMU_STRTAB_BASE_CFG.
3. Configure the base address for the Stream table by writing to SMMU_STRTAB_BASE.
4. Prevent uninitialized memory being interpreted as a valid configuration by setting STE.V = 0 for each STE to mark it as invalid.
5. Ensure that written data is observable to the SMMU by performing a *Data Synchronization Barrier* (DSB) operation.

   If SMMU_IDR0.COHACC = 0, the system does not support coherent access to memory for the TCU. In such cases, you might require extra steps to ensure that the SMMU can observe the written data.

C.1.4 Initializing the Command queue

Software must initialize the Command queue by enabling it and checking that the enable operation is complete.

To initialize the Command queue, ensure that your software performs the following steps:

Procedure

1. Enable the Command queue by setting the SMMU_S_CR0.CMDQEN bit to 1.
2. Check that the enable operation is complete by polling SMMU_S_CR0ACK until CMDQEN reads as 1.

C.1.5 Initializing the Event queue

Software must initialize the Event queue by enabling it and checking that the enable operation is complete.

To initialize the Event queue, ensure that your software performs the following steps:

Procedure

1. Enable the Event queue by setting the SMMU_S_CR0.EVENTQEN bit to 1.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

2. Check that the enable operation is complete by polling SMMU_S_CR0ACK until EVENTQEN reads as 1.

C.1.6 Invalidating TLBs and configuration caches

Before use, the MMU-700 TLBs and configuration cache structures must be invalidated by issuing commands to the Command queue. Alternatively, Secure software can invalidate all TLBs and caches with a single write.

To invalidate TLB entries, ensure that your software issues the appropriate command for the translation context. To invalidate:

- TLB entries for Non-secure EL1 contexts, issue `CMD_TLBI_NSNH_ALL`
- TLB entries for EL2 contexts, issue `CMD_TLBI_EL2_ALL`
- TLB entries for EL3 contexts, issue `CMD_TLBI_EL3_ALL`
- TLB entries for Secure EL1 contexts, issue `CMD_TLBI_NH_ALL`

Note

Commands to invalidate Secure TLB entries can only be issued through the Secure Command queue. For a system that implements two Security states, Secure software must issue the appropriate command to the Secure Command queue for the first TLB invalidation. If your system does not use Secure software, you can permit Non-secure software to access SMMU_S_INIT by using either **sec_override** or the *3.7.7 TCU_SCR register* on page 3-112.

To invalidate both the TCU configuration cache and the TBU combined configuration cache and TLB, issue the `CMD_CFGI_ALL` command.

To force all previous commands to complete, issue `CMD_SYNC`.

To invalidate all configuration caches and TLB entries for all translation regimes and Security states, ensure that Secure software:

1. Sets SMMU_S_INIT.INV_ALL to 1. The SMMU sets SMMU_S_INIT.INV_ALL to 0 after the invalidation completes.
2. Polls SMMU_S_INIT.INV_ALL to check it is set to 0 before continuing the SMMU configuration.

For more information about issuing commands to the Command queue, see the *Arm® System Memory Management Unit Architecture Specification, SMMU architecture versions 3.0, 3.1 and 3.2*.

C.1.7 Creating a basic Context Descriptor

A *Context Descriptor* (CD) is a data structure in system memory. A CD defines how Stage 1 translation is performed. The SubstreamID is used to select the CD.

To create a CD, ensure that your software performs the following steps:

1. Allocate 64 bytes of memory for the CD.
2. Configure the CD fields according to the information in the following table.

Table C-1 Configuring the CD

Field	Description
AA64	Translation table format: **0** AArch32. **1** AArch64.
EPD0	Enable translations for TTB0 by setting EPD0 to 0.
TTB0	Base address of translation table 0.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Table C-1 Configuring the CD (continued)

Field	Description
TG0	Translation granule size for TTB0 when CD.AA64 = 1.
IR0 OR0	Cacheability attribute to use for translation table walks to TTB0: **00** Non-cacheable. **01** Write-Back Cacheable, Read-Allocate Write-Allocate. **10** Write-through Cacheable, Read-Allocate.
SH0	Shareability of translation table walks to TTB0: **00** Non-shareable. **01** Outer Shareable. **10** Inner Shareable.
EPD1	If the StreamWorld supports split address spaces, enable table walks for TTB1.
ENDI	The endianness for the translation tables.
IPS	The IPA size when CD.AA64 = 1.
ASET	Defines whether the ASID values are shared with the ASID values of an Arm processor. —— **Note** —— If you expect this context to receive broadcast TLB invalidation commands from a PE, set ASET to 0. ——
V	Valid CD. This field must be set to 1.

C.1.8 Creating a Stream Table Entry

Each *Stream Table Entry* (STE) configures how Stage 2 translation is performed, and how the *Context Descriptor* (CD) table can be found. The StreamID is used to select an STE.

To create an STE, ensure that your software performs the following steps:

1. Allocate 64 bytes of memory for the STE.
2. Set the STE.Config field as required for Stage 1 translation, Stage 2 translation, or translation bypass:

   `0b000` No traffic can pass through the MMU. An abort is returned.

   `0b100` Stage 1 and Stage 2 bypass.

   `0b101` Stage 1 translation Stage 2 bypass.

   `0b110` Stage 1 bypass Stage 2 translation.

   `0b111` Stage 1 and Stage 2 translation.
3. If Stage 1 translation is enabled, you can set the following fields:

   STE.S1CDMax Controls whether STE.S1ContextPtr points to a single CD or a CD table.

   STE.S1Fmt If STE.S1CDMax > 0, configures the format of the CD table.

   STE. S1ContextPtr Contains a pointer to either a CD or a CD table. If Stage 2 translation is enabled, this pointer is an *intermediate physical address* (IPA), otherwise it is an untranslated *physical address* PA.
4. If Stage 2 translation is enabled, you can set the following fields:

   STE.S2TTB Points to the Stage 2 translation table base address.

   STE.S2PS Contains the PA size of the stage 2 PA range.

   STE.S2AA64 Indicates whether the Stage 2 tables are AArch32 or AArch64 format.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

STE.S3ENDI	Set this field to the required endianness for the stage 2 translation tables.
STE.S2AFFD	Disable Access Flag faults for Stage 2 translation.
STE.S2TG	`0b00`: 4KB. `0b01`: 64KB. `0b10`: 16KB.
STE.S2IR0 and STE.S2OR0	`0b00`: Non-cacheable. `0b01`: Write-Back Cacheable, Read-Allocate Write-Allocate. `0b10`: Write-through Cacheable, Read-Allocate.
STE.S2SH0	`0b00`: Non-shareable. `0b01`: Outer Shareable. `0b10`: Inner Shareable.
STE.S2VMID	Contains the VMID associated with these translations.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

C.2 Enabling the SMMU

Software can enable the SMMU by writing to SMMU_CR0 after the Stream table is populated.

To enable the SMMU, carry out the following procedure.

Procedure

1. Ensure that all Stream table entries are populated in memory.
2. Set the SMMU_CR0.SMMUEN bit to 1.
3. Check that the enable operation is complete by polling SMMU_CR0ACK until SMMUEN reads as 1.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

Appendix D
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

- *D.1 Revisions* on page Appx-D-224.

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential

D.1 Revisions

This appendix describes the technical changes between released issues of this book.

Table D-1 Issue 0000-01

Change	Location
First release	-

Table D-2 Differences between issue 0000-01 and issue 0000-02

Change	Location
Improvements to descriptions	Throughout the document

Table D-3 Differences between issue 0000-02 and issue 0001-03

Change	Location
Improvements to descriptions	Throughout the document
Added new parameters	• *2.5.2 TCU buffer configuration options* on page 2-75 • *2.5.8 TBU buffer configuration options* on page 2-81
Added system discovery registers	• *3.9 TCU system discovery registers* on page 3-120 • *3.16 TBU system discovery registers* on page 3-153

Table D-4 Differences between issue 0001-03 and issue 0001-04

Change	Location
Improvements to descriptions	Throughout the document
Added a 'Width' column to all signal description tables	*Appendix A Signal descriptions* on page Appx-A-169

Copyright © 2019–2021 Arm Limited or its affiliates. All rights reserved.
Non-Confidential